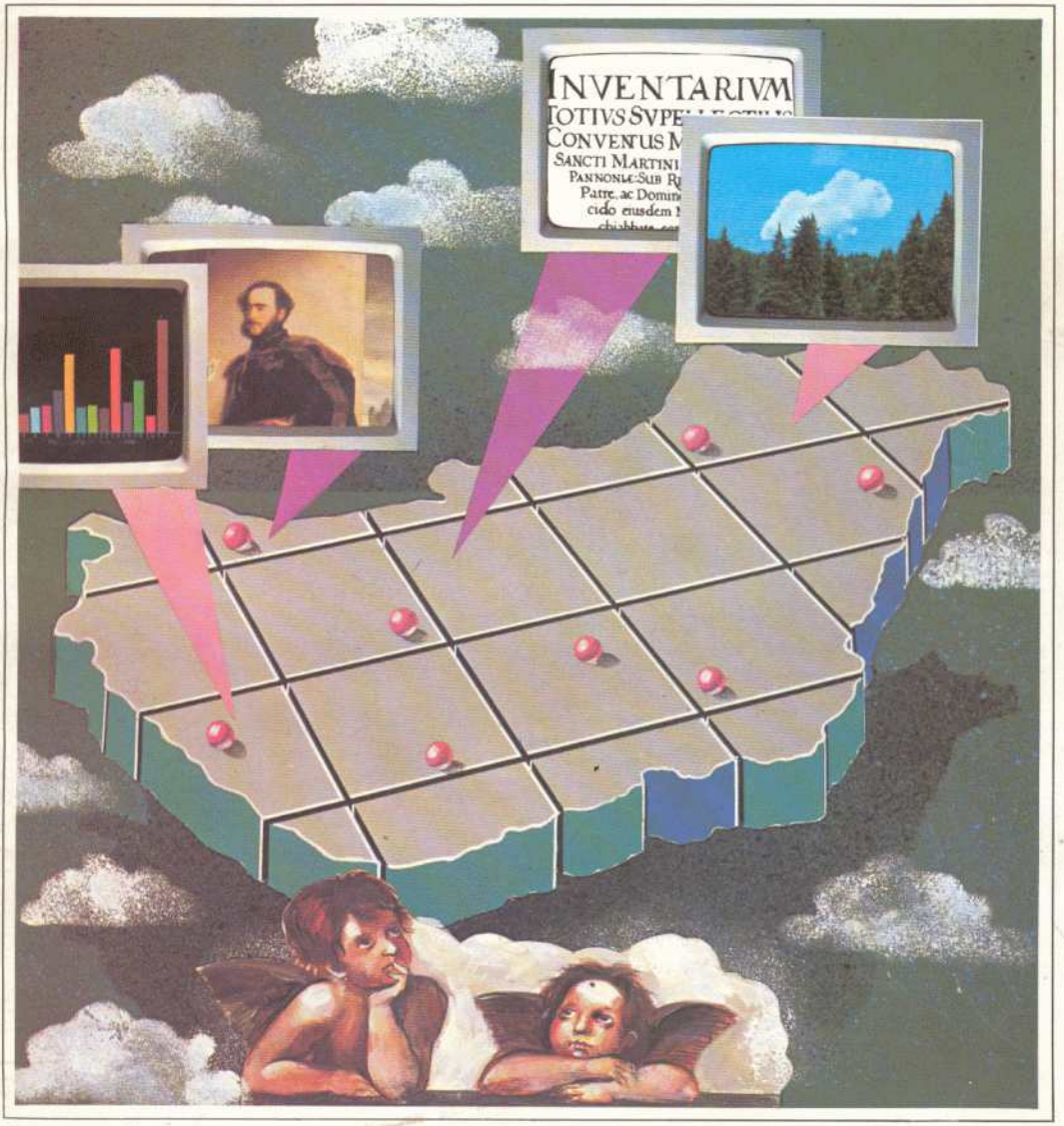
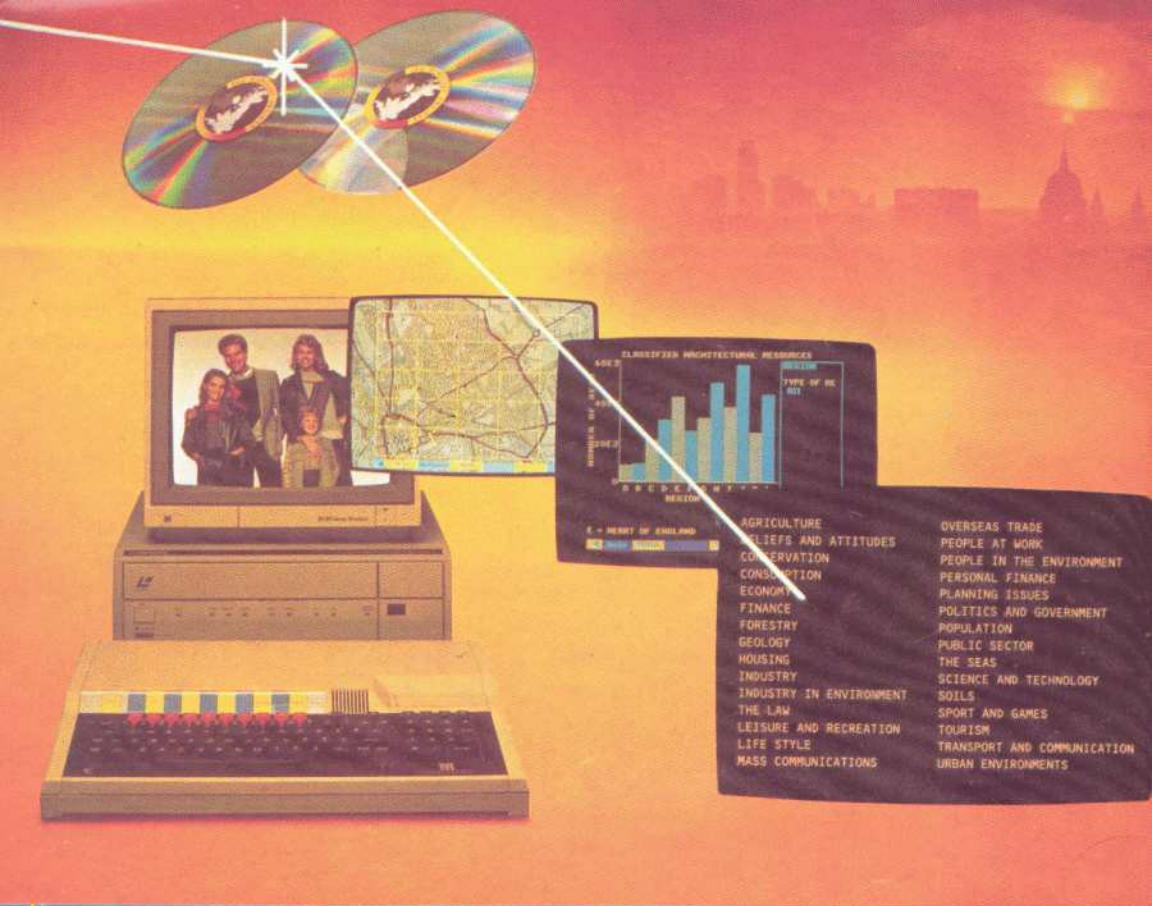
Ára: 30 Ft

mikro számítógép magazin



ÁLOM, ÁLOM...

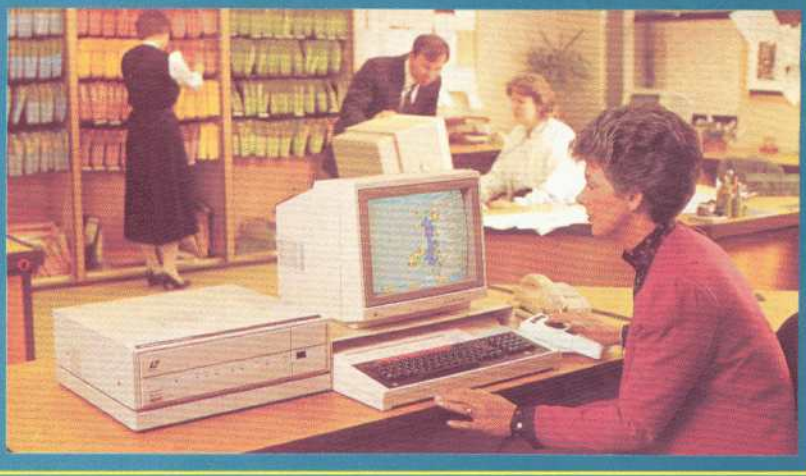
1988/2



Britannia, óh!

sóhajtanak fel címlapunk angyalkái. Ezt teheti velük együtt sok érdeklődő is, akik szívesen „járnák" be számítógépük monitora előtt ülve például a pécsi Vasarely-kiállítás termeit.

Az angoloknak ez már valóság, hiszen van Domesday rendszerük, amelyről e számunk 26. oldalán olvashatnak. Reméljük, hogy egy hasonló hazai rendszerre nem kell ítéletnapig várnunk!



6. ÉVFOLYAM
1988/2. SZÁM

μ mikro számítógép magazin

A NEUMANN JÁNOS SZÁMÍTÓGÉPTUDOMÁNYI TÁRSASÁG ÉS A KISZ KÖZPONTI BIZOTTSÁG LAPJA

A kiadvány a Tudományszervezési és Informatikai Intézettel együttműködve készül

A szerkesztőbizottság vezetője:
Kovács Győző
A szerkesztőség munkatársai:
Babos János (tervezőszerkesztő)
Bakos Tamás (programozástechnika)
Broczkó Péter (hírek)
Énekes Ferenc (KISZ)
Jakab Ágnes (olvasószerkesztő)
Kovács Győző (levelezés)
Krasznai Éva (Diákszerkesztőség)
Lindner László (sakk)
Petróczy Judit (könyvek)
Pinke György (NJSZT, alkalmazások)
Simonyi Endre
Szebenszki Sándor
Szulyovszky Csaba
Tamásné Lakó Erika
Terebessy Ákosné
Varga András (TII, iskola-számítógép)
Vizessy Mária

Címképünk:
Ramocsai Imri munkája

Felelős szerkesztő:
Könyves Tóth Pál

Szerkesztőség:
1027 Budapest, Fő u. 68.
Telefon: 154-250

Levélcím:
1371 Budapest
Pf. 433.

Kiadja az Ifjúsági Lap- és Könyvkiadó Vállalat

Felelős kiadó:
dr. Király G. István
igazgató

Kiadóhivatal:
1065 Budapest, Révay u. 16.
Telefon: 116-660

Terjeszti a Magyar Posta Előfizethető a hírlapkézbesítő hivataloknál és a Posta Hírlapelőfizetési és Lapellátási Irodáján (1900 Budapest V., József nádor tér 1.) vagy átutalással a 215—96 162 pénzforgalmi jelzőszámra.

Megjelenik havonta
Egy szám ára 30,— Ft
Előfizetési díj:
egy évre 360,— Ft
fél évre 180,— Ft
Külföldön terjeszti a Kultúra,
1389 Budapest, Pf. 149.
és a Magyar Média
1932 Budapest, Pf. 279.
86—0253

Szikra Lapnyomda
Budapest (88—1999)
Felelős vezető:
Csöndes Zoltán vezérigazgató

INDEX: 25 629
ISSN 0236-6088

Tartalom

Egy diákversenyről, nagyon komolyan 2
Számítástechnikai és rokon szakmák 10
Adok-veszek-cserélek 23
SOFTWARE '88 25
A Domesday-rendszer 26
Az év számítógépei 28
Miért nem stupid a MUPID? 29
Újföld 31
μINFORM 36
Az alapoktól a távoktatásig 41
Olvastunk . . . 42

ISKOLA-SZÁMÍTÓGÉP

TechnoMIR 3
RESET-védelem 4
Perifériák közös használata 5

DIÁKROVAT

Függvényábrázolás és integrálszámítás C16-on 6
Sejtszaporodás C64-en 7
Örökélet C16-ra 7
Egy kis zene a Plus/4-en 8
Egy apró, de érdekes program C64-re 9

PROGRAMOZÁSTECHNIKA

BASIC és gépi kód 13
Z80 programok haladóknak Spectrumra és Primóra 14
Számítógépes grafika Pascalban. Bezier-görbék 17
Függvények és utasítások 20
Bináris fák 21
A HELP és a SUPERMONITOR együttes működtetése 24

μPROGRAMOK

Képernyőmásolás 33

μKLUB

A SMARTWORK 37
Primós harc 40

SAKK

A játékfa és kiértékelése 44

AZ OLVASÓ ÍRJA 45

KÖNYVEK 46

HÍREK, ÉRDEKESSÉGEK 47

PONTVADÁSZAT 48

Egy diákversenyről, nagyon komolyan

„Rendezzünk autóversenyt! A részt vevő autók száma ötnél ne legyen kevesebb, tíznél pedig ne több! A sebességüket véletlen számok előállításával a gép határozza meg. Egyenes pályán mozognak, amelynek a végén van a cél. A gép figyelje a célba befutók sorrendjét, és a verseny végén írja ki az első három helyezett sorrendjét! Minden más (névadás, pályahossz) a te kedvedre van bízva. Csak arra vigyázz, hogy a fenti előírásoknak programod mindenképp megfeleljen!"
(Versenyfeladat a vak és gyengénlátó általános iskolások első országos versenyére.)

Elárulom, van sok, de egy rendkívül gyenge oldalam: ha olyan eseményre kapok meghívást, ami bármilyen kapcsolatban van az ifjúsággal, nem tudok nemet mondani.

Így jártam októberben a pécsi Medisoft '87 kiállítással, és a két nappal később kezdődött orvosi informatikai vándorgyűléssel is. Miután az eseménysorozat nyitányaként egy diákprogramozói versenyt is rendeztek, nem tudtam a felkérést visszautasítani, és elvállaltam a zsűri vezetését.

Azt is tudtam, hogy ez a verseny nagyon különbözni fog minden eddigitől, mert először az országban, külön kategóriában ugyan, de az általános iskolások, a középiskolások és az egyetemisták mellett világtalan és gyengénlátó gyerekek is versenyeznek.

Az ötlet szülőatyja Hozman József volt, a kiállítás fő szervezője, az EISZI főosztályvezetője, a megvalósítói pedig elsősorban a Vakok és Gyengénlátók Országos Szövetségének munkatársai, és közülük is leginkább Csák Valéria, aki élharcosa a látáskárosult gyerekek számítástechnikai oktatásának. Hogy el ne felejtsem, az esemény jelentőségét mutatja, hogy a versenyre eljött a szövetség főtitkára, dr. Bódi István és helyettese, Daru Gerőné is, akik nemcsak protokolláris okokból voltak a verseny egész tartama alatt a gyerekekkel, hanem azért is, mert felelősséget éreznek a vak gyerekek számítástechnikai oktatásáért, ami talán új és nagyon reményteljes utat nyit e gyerekek értékes felnőtté válásához.

A versenyre három iskola küldte el csapatát: a budapesti, valamint a debreceni gyengénlátók és a budapesti vakok általános iskolája. A verseny zsűrije mind az egyéni, mind pedig a csapatteljesítményt értékelte. A kép teljességéhez hozzátartozik, hogy ezek a gyerekek még csak kilenc hónapja tanulnak BASIC-ben programozni a dombóvári Color Ipari Szövetkezet gyártotta Brailab gépen. Ha valaki nem tudná, egy beszélő Homelab gépről van szó, amelyet a Lukács fivérek terveztek, és amelyhez a beszélő szoftvert Arató András és felesége, Vaspöri Teréz, a KFKI munkatársai fejlesztették ki. (Emlékeztetjük az olvasót 1987/8. számunk Hallásolvasás című írására. — A szerk.)

A mottóként idézett versenyfeladatot én túlzottan nehéznek éreztem, de a gyerekeket kísérő szakembereknek az volt a véleményük, hogy aggodalmam felesleges, csak figyeljem az eseményeket. Figyeltem. Igazuk volt. Ha a versenyt nem látom, nem hiszem el, hogy ilyesmi létezik.

A versenyző gyerekek számítógép-kezelési technikában alig maradtak el a látóktól. Úgy írták egymás után az utasítássorokat, hogy nem győztem a számat tátani. Nyilvánvalóan a gyengénlátó gyerekek nemcsak hallották a programot, de rettentő közelről nézve a képernyőt látták is, persze a rövid látótávolság miatt a képernyőnek csak egy nagyon kis részét, amelyet beszűkült látómezejük befogadott. Megfeszített figyelemmel írták a programokat, és a fejük a felírás sebességével követte a szöveget a képernyő előtt. A fejek együtt mozogtak a szöveggel, és a program egyre jobban közelített a megoldáshoz.

Az egyik teljesen világtalan gyerek programhibát vétett, és elbámultam, ahogyan elkezdte a programot javítani. Gyorsan futtatta a képernyőn a sorokat, feltételezem, hogy az agyában ugyanigy futott a programok képe is. A szinkronitás annyira tökéletes volt, hogy a programot pontosan ott állította meg, ahol a hibás sor volt, meghallgatta a képernyőre felírt feliratot, és egy pillanat alatt kijavította a tévedését. Ismétlem, ha nem látom, nem hiszem, és mindezt kilenc hónap tanulás után.

Nagyon nehéz leírni a verseny hangulatát és azt az érzést, amely az embert ezeknek a rendkívüli intenzitással dolgozó gyerekeknek a láttán elfogta. Meg sem próbálom. A gyerekeket kísérő nevelők elmondták, de én is így láttam, hogy amikor egy világtalan vagy gyengénlátó gyerek a számítógépen dolgozik, akkor éppen a látvány hiánya miatt, rendkívüli módon kell koncentrálnia. Így ellentétben a szokásos versenyekkel, itt gyakorlatilag nem volt szükség felügyelőkre, hiszen minden gyerek a saját világába zártan csak a saját gondolataira koncentrált, és eszébe sem jutott, hogy „puskázzon". Még mielőtt bárki is föltételezné, hogy éppen ezért a versenyző gyerekek talán nem is alkottak közösséget, azokat sietve felvilágositom, hogy ellenkezőleg, soha ilyen összetartó gyerekeket én nem láttam. Amikor kihirdettem az eredményt, a helyezetteket a többiek úgy ünnepelték, mintha egy világversenyt nyertek volna meg, nem lehetett meghatódás nélkül nézni azt az odaadó örömet, amely az arcukról sugárzott.

Nem tudom, hogy máshol, például külföldön rendeztek-e ilyen versenyeket, úgy hallottam, hogy nem. Már ezért is föltétlenül folytatni kellene ezt a kezdeményezést. Talán nem túlzás azt mondani, hogy önmagunkat becsüljük meg azzal, ha a számítástechnikát elérhetővé tesszük a vakok és gyengénlátók számára is, akiknek — eddig javarészt — inkább az olyan közügyességet igénylő pályák voltak úgy-ahogy nyitottak, mint a kosárfonás, a telefonközpont-kezelés, a maszszőri mesterség, a szellemi pályák közül pedig a jogászság. Az informatika a tehetséges gyerekeknek nem is csak egy ajtó, hanem egy nagyon széles kapu, amelyen keresztül a gazdaság sokféle területén tudnak majd érvényesülni.

Nem szeretném — ezzel a nagyon optimistára sikerült beszámolóval — a még bőven meglévő problémákat elkendőzni, mert azok persze vannak. Az is igaz viszont, hogy ezeket a problémákat egyáltalán nem lehetetlen megoldani. Szerintem ugyanis a számítástechnikának a vakok és gyengénlátók iskolájában vagy a felnőttek részére speciális SZÁMALK-tanfolyamokon való oktatása csak a társadalmi feladat felének a megoldását jelenti. Igaz, hogy a tudás hatalom, de ez a hatalom csak akkor ér valamit, ha élni is lehet vele. Megfordítva a mondatot: ez a tanitási, tanulási erőfeszítés akkor fogja elérni a célját, ha erre a tudásra a társadalomnak is szüksége van, és ebből a tudásból — akik megszerezték — meg is tudnak élni, képesek lesznek a tanultak alkalmazásával a látó emberekhez hasonló módon pénzt is keresni.

Az iskolásoknak néhány év múlva, a felnőttként tanulóknak néhány hónap múlva munkahelyre lesz szükségük, mert különben ismereteik öncélúak maradnak, gyakorló programokat írhatnak életük végéig, olyanokat, amelyeket senki sem tud felhasználni.

Véleményem szerint *társadalmi összefogásra van szükség*, a társadalomba persze beleértem az állami adminisztrációt is, hogy legalább a nagyobb számítástechnikai cégeknél létrejöjjenek speciális munkahelyek, pontosan olyanok, mint amelyeket az intézetek munkatársai is használnak. Azt hiszem, hogy IBM kompatibilis számítógépekre van szükség, speciális perifériákkal (például Versabraille) felszerelve, esetleg az IBM PC-re átdolgozott Brailab modullal kiegészítve. Egy ilyen munkahely nem olcsó, és nagyon jól tudom, hogy ma a vállalatok meggondolják, mire adják ki a pénzüket. Ennek ellenére úgy vélem, hogy ezen nem szabad spórolni a vállalatoknak vagy a pénzügyi kormányzatnak. Az említett speciális perifériák sajnos egyelőre kizárólag tőkés importból szerezhetők be. Valószínűleg nem azon múlna az ország devizamérlegének a hiánya, hogy néhány tiz vagy néhány száz ilyen berendezést importálunk, de ezen múlik néhány száz világtalan honfitársunk sorsa, ami — ugye igazat adnak nekem — azért mégiscsak fontosabb valamennyiünknek, mint bármi más a világon.

Tudom, hogy a µM nem a nagy politikát csináló fórum, és nem is olyan lap, amelyet a honatyák komoly számítástechnikai döntéseik előtt föltétlenül fellapoznak. Mégis azt remélem, hogy felhivásunknak lesz foganatja, és jelentkeznek olyan intézmények, amelyek úgy döntenek, hogy szivesen létesitenek a vakok részére munkahelyeket. Mi vállalkozunk arra — úgy is, mint a Neumann Társaság —, hogy a jelentkezéseket összegyűjtjük és az illetékesek elé visszük döntésre. Szivesen koordináljuk a fejlesztési munkát is, hogy lehetőség szerint uniformizált és ne különböző egyedi megoldások szülessenek. Ez különösen azért fontos, mert akkor a vak programozó akárhol is dolgozik, mindenhol mindent ugyanott talál, nem beszélve a szoftverkörnyezetről, amelyet nem kell mindig újra megtanulni.

Persze az igazi megoldás az volna, ha sikerülne olyan olcsó hardvert találnunk, ami otthoni munkahelyek kialakítását tenné lehetővé, mert akkor az intézetekbe való bejárás nélkül tudnák a vak emberek állandó foglalkoztatását biztositani. Talán a nem túl távoli jövőben erre is sor kerülhet.

Addig is várjuk a vakok és gyengénlátók ügyét magukénak érző intézmények jelentkezését.

KOVÁCS GYŐZŐ

TechnoMIR

Egy pontos óra

A legfontosabb digitális egységek után most áttértünk a bonyolultabb felépítésű és/vagy nehezebben kezelhető részekre. Ezek mindegyike valamilyen speciális igény kielégítésére készült.

A független áramkörű óra (hardware clock) se nem igazán független, se nem kizárólag óra. Az viszont igaz, hogy csak Z80 alapú gépekhez használható.

A modul lelke egy U 857 típusú számlálóidőzítő áramkör, így számlálónak, időzítőnek, órának, frekvenciamérőnek stb. használható az eszköz. A belső struktúra az *ábrán* látható. Érdekes megoldás, hogy a négy, részben független számláló sorba van kötve (óraüzem), de megfelelő csatlakozókon a be- és kimenetek külön-külön is elérhetők.

A négy csatorna külön-külön programozandó OUT, illetve POKE utasításokkal, és külön-külön olvasandó is (INP, illetve PEEK). Az egyes csatornák HT címei:

I. 32
II. 33
III. 34
IV. 35

Az óra üzemmódjai a következők.

Számláló üzemmód

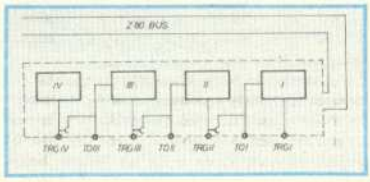
Ebben az üzemmódban a csatorna TRG bemenetére adott impulzusokat számlálja. A számláló az érkező aktív élre a rendszerórához szinkronizálva dekrementálódik. A számlálás megkezdése előtt felveszi a konstans regiszter értékét.

A bemeneti vezérlő él és az órajel felfutó éle között nem szükséges előkészítési idő, de a számláló értéke mindig csak a következő órajellel csökken. A külső bemenetre az aktív él programozható.

Az I., II. és III. csatornánál, ha a számláló elérte a nullát, a csatorna kimenete aktív állapotba kerül. A lefelé számláló közben felveszi a konstans regiszter tartalmát anélkül, hogy a számlálás megszakadna. Ha a számlálás tartama alatt a konstans regiszter megváltozik, az új értéket a számláló csak a számlálás befejezése után veszi fel.

Időzítő üzemmód

Ebben az üzemmódban a rendszeróra két sorba kötött számlálóra (előszámláló és lefelé számláló) kerül, és így a csatorna időintervallum előállítására alkalmas. Az előszámláló programozhatóan 16-tal vagy 256-tal osztja a rendszer órajelét. Az előszámláló kimenete a lefelé számláló bemenetére csatlakozik, ami a konstans regiszter tartalmával, mint kezdeti értékkel tölthető fel. A számláló nullátmeneténél a csatorna



kimenetén (T0) egy impulzus jelenik meg, miközben a számláló felveszi a kezdő értéket (konstans regiszter). A két impulzus közötti idő a következőképpen számolható ki:

$t \times P \times TC$

ahol t — a rendszer órajelének periódusideje
P — az előszámláló osztása (16 vagy 256)
TC — a beprogramozott konstans

Az időzítés indítását vagy a betöltést követő rendszerórajel végzi, vagy a TRG bemenet aktivizáló éle. Ez programozható.

A modul programozásának menete a következő.

El kell dönteni, hogy az egyes csatornákat milyen módban használjuk: mint számláló vagy mint időzítő. Létre kell hozni a megfelelő kapcsolatot a modul és a külvilág, illetve az egyes csatornák között. Be kell programozni a csatornákat a megfelelő címre töltött vezérlő- és konstans regiszterekkel. Ez lehet a felhasználói program eleje is, de parancsként is kiadhatók a megfelelő utasítások. Végül el kell indítani a felhasználói programot a számítógépen.

A vezérlőregiszter értéke a *táblázat* felhasználásával kiszámítható. Ezt kell a csatorna címén legelsőnek kiküldeni. Ezt követi — ha kell, hogy kövesse — a konstans regiszter értéke. Ez egy 0—255 intervallumba eső egész szám lehet. A csatorna működés közben bármikor olvasható, az eredmény a számláló aktuális értéke.

Vigyázzunk! A csatornába írt adatot a z vezérlőregiszter értéknek tekinti, és ennek megfelelően reagál. Más szóval a csatorna programja bármikor átírható egyetlen paranccsal.

Végezetül egy egyszerű programot közlünk a *listán* HT—1080Z számítógépre. A program indítása után az indítástól számított idő íródik a képernyőre, mindaddig, amíg egy billentyűt le nem nyomunk.

Jó munkát kívánunk mindenkinek, és várjuk a leveleket.

ALBU LÁSZLÓ—KIRÁLY LÁSZLÓ

```
    ****************************
    *                          *
    * PONTOS ÓRA HT 1080Z-RE   *
    *                          *
    ****************************

    ****************************
    *     BEALLITASOK          *
    ****************************

    * PERC SZAMLALO IV. CSAT. *
10  OUT 35,69 : OUT 35,60
    * MPERC SZAMLALO III. CSAT. *
20  OUT 34,69 : OUT 34,60
    * SZAZAD MP SZAMLALO II. CSAT. *
30  OUT 33,69 : OUT 33,100
    * SZAZAD MP IDOZITO I. CSAT. *
40  OUT 32,37 : OUT 32,70

    ****************************
    *     FOPROGRAM            *
    ****************************

50  PRINT@0,60-INP(35);60-INP(34);
60  PRINT 100-INP(33)
70  IF INKEY$="" THEN 50
80  END
```

A vezérlőregiszter bit szerinti jelentése

A bit sorszáma	A bit értéke	A bit jelentése
7.	1	A megszakítás engedélyezett
7.	0	A megszakítás tiltott

* Nincs bekötve, a bit értékét tartsuk 0-ban

6.	1	Számláló üzemmód kijelölve
6.	0	Időzítő üzemmód kijelölve
5.	1	Előszámláló osztása 256
5.	0	Előszámláló osztása 16

* Csak időzítő üzemmódban van előosztás

4.	1	A felfutó él az aktív
4.	0	A lefutó él az aktív

* Időzítő üzemben a külső TRG jel aktív élére indul az időzítés
* Számláló üzemben az órajel aktív élére dekrementálódik a számláló

3.	1	Időzítés külső TRG jellel indítható
3.	0	Az időzítést a rendszer órajele indítja

* Csak időzítő üzemmódban
* Az indítás csak a konstans regiszter feltöltése után érvényesül

2.	1	A következő adat a konstans regiszterbe kerül
2.	0	Érvényes a konstans regiszter régi tartalma

* Érvényes konstans regiszter nélkül a csatorna nem működik!

1.	1	RESET, a számlálás, ill. időzítés leáll
1.	0	A csatorna folytatja a működést

* Ha a 2. és 1. bit is 1, a csatorna konstans regiszter betöltést vár
* Ha csak az 1. bit 1, a csatorna leáll, de nem változik a regiszterek tartalma (1. bit 0= CONTINUE)

0.	1	Az adat vezérlőregiszter-érték
0.	0	Az adat nem vezérlőregiszter-érték

PLUS/4

RESET-védelem

C64-esen ugyanezt úgy oldottam meg, mintha egy BASIC ROM-bővítést tettem volna a bővítő csatlakozóba. Bizonyára mindenki tudja, hogy ezeknek a bővítéseknek valamilyen módon át kell adni a vezérlést. Ezt a vezérlésátadást az alapgép bekapcsolásakor egy, a KERNAL-ban levő rutin automatikusan végrehajtja, mégpedig a következő módon.

Az FCE7 címen található egy JSR FD02 utasítás. Ez az FD02-es rutin ellenőrzi, hogy van-e ROM-bővítés vagy nincs. Az FD10-es címtől kezdve a következő karaktersorozat található: 'CBM80', amely hexadecimálisan így néz ki:

FD10: C3 C2 CD 38 30

Ennek kell a 8004-es címtől kezdődően lennie egy ROM-bővítés esetén. Ha ott van, akkor a rutin, értesülve a bővítésről és a vizsgáló rutinból való visszatérés után elugrik a 8000—8001-es bájtokon található címre. Itt annak a rutinnak a kezdőcíme kell hogy álljon, amelyik ezt a bővítést inicializálja. Ha nincs ott a keresett karaktersorozat, folytatódik az eredeti inicializáló rutin.

A ROM-bővítést szimulálni is lehet. A C64-es esetén a következőképpen: 8004-től kezdődően beírom a 'CBM80' karaktersorozatot. 8000—8001-re az FCE 2-es címet, 8002—8003-ra pedig egy NMI (nem maszkolt megszakítás) rutin kezdőcímét, ami lehet az eredeti NMI rutin címe is: E37B. (Természetesen a megszokott alsó-felső bájt sorrendben.)

Ezek után ha valaki a soros buszon vagy a USER porton RESET-et old ki, a gép végtelen ciklusba kerül, ugyanis a RESET rutin kezdőcíme (mindig) FCE2, és mivel az ellenőrző rutin 8004-től ott találja a 'CBM80'-at, elugrik a 8000-en található címre, vagyis vissza az FCE2-re, és kezdi elölről. Ebből a STOP/RESTORE gombok együttes megnyomása után még ki lehet szállni, ugyanis az E37B címen levő NMI rutin figyeli ezt a gombpárt. Ha a 8002—8003-ra is például az FCE2 címet írom, akkor végleg bezárult a kapu. Ezután kilépni csak ki/bekapcsolással lehet, vagy ha valakinek ténylegesen van egy ROM-bővítése a csatlakozóban.

Ez az út a Plus/4-es esetében sajnos több szempontból is járhatatlannak bizonyult.

Először is nem találtam ilyen ROM-bővítést ellenőrző rutint (ami nem jelenti azt, hogy nincs is). Másodszor a Plus/4-esen nemcsak egyszerű RESET-et lehet csinálni, hanem ha a STOP gomb nyomvatartása mellett nyomom meg a RESET gombot, akkor egy, a ROM-ban levő monitorral — a TEDMON-ra — adódik át a vezérlés. Ennél a gépnél azonban — szerencsére (?) — nincs RESTORE gomb.

```
:4000 LDX  00
:4002 LDA  8000,X - 8000-FD00 közt
:4005 STA  8000,X - levő ROM átmáso-
:4008 INX         - lása az alatta
:4009 BNE  4002   - levő RAM-ba.
:400B INC  4004
:400E INC  4007
:4011 LDA  4007
:4014 CMP  FD
:4016 BCC  4002
:4018 LDX  49     - FF49-FFFF közt
:401A LDA  FF00,X - levő ROM átmáso-
:401D STA  FF00,X - lása a RAM-ba.
:4020 INX
:4021 BNE  4017
:4023 LDA  80     - 8000-es cím le-
:4025 STA  FFFD   - pakolása az FFFC
:4028 STX  FFFC   - címre.
:402B STX  8001   - 8000-es JMP címé-
:402E INX         - nek átírása.
:402F STX  FF3F   - átkapcsolás RAM-ra
:4032 RTS
```

1. lista

2. lista

```
:402B LDX  startcím   - a program start-
:402D STX  8001       - címének lepako-
:4030 LDX  startcím   - lása alsó-felső
:4032 STX  8002       - byte sorrendben
:4035 LDX  01
:4037 STX  FF3F
:403A RTS
```

A következő megoldást találtam. Mivel rendelkezésemre állt a 64 kbájt RAM, ezért az egész ROM-ot átmásoltam az alatta levő RAM-ba. Majd az átmásolt KERNAL két címét átírva elértem, hogy a monitor helyett is a 8000-es címre adódjon át a vezérlés. Ott egy JMP utasítás található címével együtt. (Ellentétben a C64-essel, ahol itt csak egy címnek kell állnia, JMP utasítás nélkül.) Ha ennek az utasításnak a címét például önmagára írom:

:8000 4C 00 80 JMP 8000

akkor ez egy végtelen ciklust eredményez a RESET gomb megnyomásakor.

Valamivel elegánsabb, ha oda egy olyan rutinunknak a címe kerül, amelyik például az FFD2 KERNAL CHROUT rutin segítségével kiírja, hogy: NANA! vagy: EZ NEM SZÉP DOLOG! stb., majd a kiírás után egy JMP utasítással önmagára ugrik vissza. De lehet ez a cím a programunk startcíme is. Akkor pedig a RESET megnyomásakor a program futása elkezdődik elölről.

Az *1. listán* látható az a rutin, amely elvégzi a ROM átmásolását, átkapcsol ROM-ról RAM-ra, és átírja a címeket. Egy megoldás látható a startcímes esetre a *2. listán*. 402B-ig megegyezik az 1. rutinnal.

Ez a rövid rutin bármely programhoz hozzáfűzhető a TEDMON segítségével is. SAVE-eléskor ügyelni kell, hogy ezt a hozzáfűzött részt is kimentsük. Célszerű monitorból kimenteni az egész programot.

RESET-védelmünk még javítható, ha ezt a rutint a program végére tesszük. A program első utasítása az legyen, hogy meghívja ezt a rutint (beállítja a védelmet), majd folytatódhat maga a program.

Még ennél is lehet jobb, ha van valakinek olyan magnóturbója, amelyik a programot autostartossá (betöltés után azonnal indulóvá) teszi. Ennél ha valaki betöltés közben lövi le a programot, vállalja azt a rizikót, hogy nem töltődik be teljesen a program (ami valószínű is). Ha pedig már betöltődött, akkor a program első dolga az, hogy ezt a védelmet feléleszti. Mire észbekap, hogy fut a program, addigra a RESET-védelem már él.

WAGNER GYÖRGY

Hetet egy csapásra

Perifériák közös használata

Van szerencsém olyan tanteremben foglalkozásokat tartani, melyben 6-12 mikroszámítógép körül nyüzsögnek a tanulók. A különböző számítógép-kiosztási akciók és az ilyen-olyan források felhasználásával összeállt géppark reprezentatív mintája lehetne az iskolák számítógéptípus-variációinak.

A beszerzések a számítógépekre és jó esetben egy magnetofonra terjedtek ki. A programok, az egyre hosszabb programok azonban igénylik a megbízható, gyors programtárolást. Ugyanakkor minden géphez talán nem is indokolt (ideológiámat az anyagi helyzetem határozza meg) floppy és nyomtató beszerzése.

Az újabban MINET néven forgalmazott (a Real-Team Gmk által gyártott REALNET) lehetővé teszi a „számítógépes hálózat" kiépítését.

A vázolt kapcsolásban egy illesztőegységgel közös lemezmeghajtót és/vagy nyomtatót használunk **hat** számítógéppel. A kiépített hálózat a bejelentkezés sorrendjében biztosít a számítógépeknek hozzáférést a közös perifériához.

A hálózat lehetőségeit még messze nem használtuk ki, mert például egy, a lemezen megnyitott fájlt minden gépről el lehet érni, és így csoportos adatrögzítés valósítható meg (például szótárprogram, szókészlet feltöltése).

Jó használt vesszük viszont különböző „programfejlesztési" feladatoknál. Az azonos kiindulási feladatot minden gép behívja — és ez így, hogy hat gépet szolgál ki egymás után egy floppy, még mindig gyorsabb, mint ha mindenki egyszerre saját kazettáról hívná be ugyanazt a programot.

A legérdekesebb feladatok közé tartozik, amikor a MINET-en keresztül különböző gépeket kapcsolunk össze.

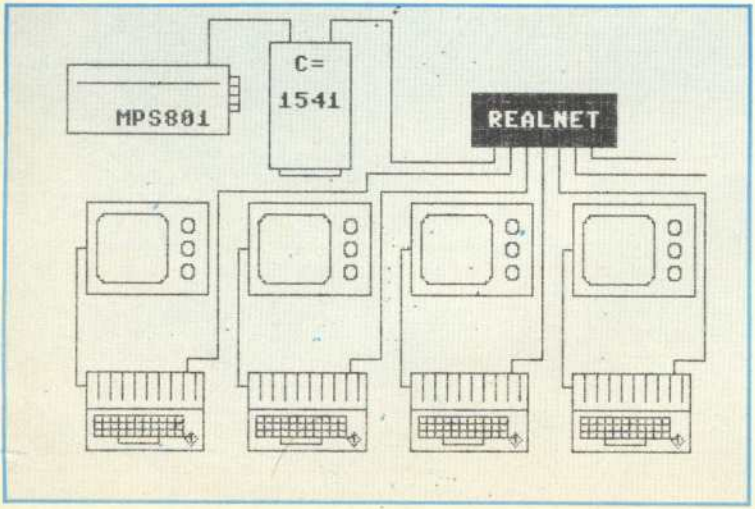
A lemezformátum azonossága miatt mód van a HT-n írt program Commodore-ba való áttöltésére és fordítva. A két gép közös elvek szerint tárolja a programokat, a programsorok szerkezete megegyező, de a kulcsszavak tokenizálása eltérő. A programokat egyik gépből a másikban futtatható formára segédprogrammal fordítjuk át. Az adatok áttöltése nem okoz különösebb gondot. Akik nem rendelkeznek a megfelelő kiépítéssel, gyakran kérik — és mi megoldjuk — a program átjátszását a fenti kapcsolás megvalósításával.

1. HT-be magnóról program (adat) beolvasása CLOAD (INPUT*)
2. Program (adat) rögzítése VC—1541-en (soros buszinterfészen keresztül)
3. MINET-en keresztül Commodore-ba program beolvasása
4. Programadjusztálás
5. Program (adat) Commodore magnóra rögzítése

Az egyszerű kezelésű interfészt többször tettük ki nyüzópróbának. Programnak lemezen való tárolása és beolvasása során — hacsak valamelyik profi Commodore program nem foglalta le a lemezmeghajtót — nem okozott fennakadást a közös periféria használata. A nyomtatós kiírásoknál viszont a megfelelő állománylezárások elmulasztásából bonyodalmak adódhatnak.

A MINET-et minden több gépet egyidejűleg használó iskolának javasolhatjuk, hiszen egy lemezmeghajtó beszerzése és az interfész 8000 forint alatti ára gazdaságos megoldást biztosít a programok megbízható és gyors tárolására. Ugyanakkor a számítógépes hálózat kialakításának elemeit is ismertetni lehet ezen keresztül. (Ez a hetedik csapás . . .)

SZ. LUKÁCS JÁNOS



A Magyar Ifjúság ALAPÍTVÁNYA a magyar ifjúság számára

A Magyar Ifjúság szerkesztősége, hosszú előkészületek és a pénzügyi fedezetek megteremtése után alapítványt hoz létre, amelyben felajánlást tesz a legkiemelkedőbb jelentőségű egyetemek számítástechnikában élenjáró diákjai számára.

Az alapítvány neve:

MAGISTER

Az alapítvány azok számára szól, akik saját egyetemük fő oktatási profiljában, illetve saját választott szakáguk profiljában a legsikeresebben és a legmegalapozottabb tudományossággal kapcsolják össze diszciplínájukat a számítástechnikával. Szóba jöhet minden olyan hardver- és szoftverfejlesztés, amely megoldja a számítástechnika újszerű bevitelét az adott területre.

Az alapítványt több évre, folyamatos működtetésre hozzuk létre. A pályázatok szándékának bejelentésére minden tanév kezdetén, október végéig nyílik lehetőség. Az elkészült pályázat benyújtási határideje a folyó tanév vége, azaz a következő év májusa. Az egyes egyetemeken alapvetően belső zsűri végzi az értékelést — némi külső szakmai képviselet egyezményes bevonásával.

Az egyes egyetemek közötti összehasonlításra, az országos első hely elérésére külön díjat írunk ki, amely az egyetemi első helyezettek pályamunkáinak elbírálása után, azok összevetésével választható ki.

Valamennyi résztvevő egyetem három díjat pályázhat. A pályadíjak összege egyetemenként a következő:

I. díj: 50 000,— Ft
II. díj: 25 000,— Ft
III. díj: 15 000,— Ft

Az országos első díj a fenti első díjhoz hozzáadott plusz 50 000 Ft, azaz 100 000 Ft.

A díjak átadására a tanévnyitókon kerül sor.

A Magyar Ifjúság egyébként széles körű szakmai és tömegkommunikációs publicitást nyújt a legjobbaknak.

A felajánlást a következő egyetemeken tesszük:

Marx Károly Közgazdaságtudományi Egyetem
Budapesti Műszaki Egyetem
Miskolci Nehézipari Műszaki Egyetem
Semmelweis Orvostudományi Egyetem
Szent-Györgyi Albert Orvostudományi Egyetem
Pécsi Orvostudományi Egyetem
Debreceni Orvostudományi Egyetem
Janus Pannonius Egyetem, Pécs
Eötvös Loránd Tudományegyetem
Kossuth Lajos Tudományegyetem
József Attila Tudományegyetem

Az alapítvánnyal kapcsolatos további kérdések tisztázása, információk nyújtása céljából javasoljuk és kérjük, hogy a Magyar Ifjúság főszerkesztője vagy annak helyettese az Egyetemi Tanács (Rektori Tanács) soros ülésére kapjon meghívást. Ez egyben azt is szolgálná, hogy az alapítvány létrehozója, a Magyar Ifjúság figyelembe venné az ott elhangzott módosító javaslatokat is.

Kérjük és javasoljuk, hogy a pályázatok meghirdetésében, a lebonyolításban vegyen részt az egyetemi KISZ-bizottság, illetve az ilyen ügyekben illetékes diákfórum is.

Függvényábrázolás és integrálszámítás C16-on

A program nem igényel hosszas magyarázatot. Betöltés és futtatás után a gép bekér egy tetszőleges függvényt. Ide bármilyen egyenletet vagy függvényt beírhatunk, de figyelni kell arra, hogy az szintaktikusan helyes legyen és hogy csak *x* lehet az ismeretlen. Ha véletlenül rossz függvényt írtunk be, a gép hibaüzenettel megáll, de nincs semmi baj, mert újraindítható. Ha megadtuk a függvényt, a program a koordinátarendszer határait kérdezi meg. Ezek után az integrálás adatait lehet beírni. Először az alsó, majd a felső határt, végül a felosztást, ami azt szabja meg, hogy a gép milyen pontosan számolja ki a függvény integrálját. Minél kisebb a felosztás értéke, annál pontosabb lesz a kapott eredmény.

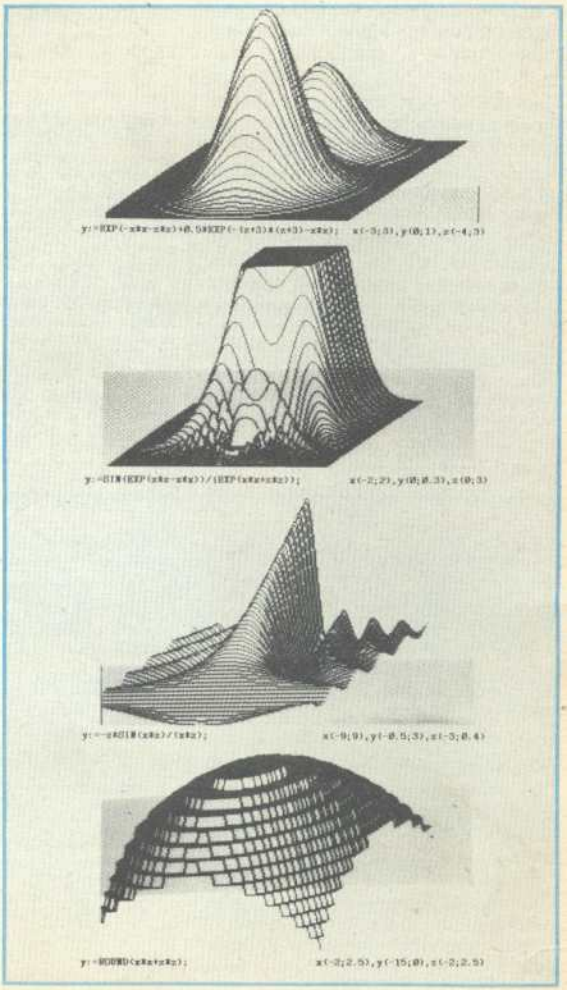
Miután minden adatot beírtunk, a gép kirajzolja a koordinátarendszert, kiszámolja az integrált (ez kis felosztás esetén elég sokáig tarthat), majd kirajzolja a függvényt, figyelembe véve a megadott határokat.

A listában a vezérlőkarakterek nem a megszokott módon vannak leírva. A könnyebb érthetőség kedvéért közöljük a jelöléseket.

SIEGLER GÁBOR

```
100 rem ******************************
110 rem *                            *
120 rem * Függvény ábrázoló és in - *
130 rem *                            *
140 rem *   tegrálszámoló program    *
150 rem *----------------------------*
160 rem *                            *
170 rem *        Készítette:         *
180 rem *                            *
190 rem *  Anonymus Software  1987   *
200 rem *                            *
210 rem ******************************
220 rem
230 graphic0,1:color0,1:color4,1:color1,2:goto 820
240 graphic 0,1
250 color0,1:color 1,2:scnclr
260 :
270 deffna(x)=sin(x)+cos(x)+1:a$="sin(x)+cos(x)+1"
280 :
290 rem
300 rem *** Értékek bevétele ***
310 rem
320 print"[clr/home][le 2][jo 2]Koordinata rendszer hatarai[le 3]"
330 input"x alsó határ  :";xa
340 input"x felső határ :";xf
350 input"y alsó határ  :";ya
360 input"y felső határ :";yf
370 if xa>=xf or ya>=yf then print"[Flash be][le]hibás
adatok[Flash ki][le 3]":goto330
380 print"[le 2][jo 2] integrálás:[le]"
390 input"  alsó határ  :";il
400 input"  felső határ :";ih
410 input"  felosztás   :";ls
420 rem
430 rem*********************************
440 rem
450 rem *** Függvényábrázolás ***
460 rem
470 scnclr:graphic 2,1
480 ifxa>0orxf<0then510
490 x1=(320/(xf-xa))*abs(xa)
500 draw 1,x1-1,0 to x1-1,159
510 ifya>0oryf<0then540
520 y1=(160/(yf-ya))*abs(yf)
530 draw 1,0,y1 to 319,y1
540 rem
550 gosub710:te=0:f=0:fort=xa to xf step((xf-xa)/320)*2
560 xh=(t-xa)*(320/(xf-xa))
570 yh=y1-(fna(t+le-38)*(160/(yf-ya)))
580 if f=0thendraw1,xh,yh:f=1:goto600
590 draw 1 to xh,yh
600 if il<t and ih>t then draw 1,xh,y1 to xh,yh
610 nextt
620 rem
630 geta$:ifa$=""then630
640 ifasc(a$)=27thengraphic0,1:end
650 goto240
660 rem
670 rem*********************************
680 rem
690 rem *** Integrálás ***
700 rem
710 te=0:forg=il to ih step ls
720 te=te+ls*fna(g+le-38):nextg
730 t1=int(te*1000)/1000
740 print"[home][le 21][jo 7],[le][ba]B[le][ba]■[&]";a$;" dx =";t1
750 print"[home][le 20]";tab(8-(len(str$(ih)))/2);ih;
760 print"[home][le 24]";tab(6-(len(str$(ih)))/2);il;:return
770 rem
780 rem*********************************
790 rem
800 rem *** Függvény bevétele ***
810 rem
820 print"[home][le 2]";
830 print"[le][jo]mi a függvény ? (x-et használj !!) "
840 input fu$
850 iflen(fu$)>24thenprint"túl hosszú kifejezés[le]":goto830
860 if len(fu$)>10theng$="[fö 4]":goto880
870 g$="[fö 3]"
880 color1,1
890 print"[clr/home][fö 6]270
deffna(x)=";fu$;":a$=";chr$(34);fu$;chr$(34)
900 print"goto 240";g$;
910 poke1319,13:poke1320,13:poke239,2
920 end
930 rem*********************************940 rem
```

```
[home]      =  'home' Karakter      ■
[clr/home]  =  'clr-home' Karakter  ■
[Flash be]  =  'Flash on' Karakter  ■
[Flash ki]  =  'Flash off' Karakter ■
[fö x]      =  x darab 'Kurzor föl' Karakter
[le x]      =  x darab 'Kurzor le'  Karakter
[jo x]      =  x darab 'Kurzor jobbra' Karakter
[ba x]      =  x darab 'Kurzor balra' Karakter
```



Sejtszaporodás C64-en

A játék a sejtek szaporodását szimulálja egy 38 × 23-as táblán. Egy sejtnek 8 szomszédja van, kivéve a tábla szélén elhelyezkedőket. A szaporodást három szabály befolyásolja:

1. Sejt akkor születik, ha egy üres cellának pontosan 3 szomszédja van.

2. A sejt akkor túlélő — a következő generációban is élő lesz —, amikor 2 vagy 3 szomszédja van.

3. A sejt valamennyi más esetben — ha 0, 1, 4, 5, 6, 7, 8 szomszédja van — elpusztul.

Először a képernyő közepén jelenik meg a nem villogó kurzor, melyet a kurzorvezérlő billentyűkkel tudunk mozgatni. Sejtet úgy helyezhetünk el a képernyőn, hogy megnyomjuk a SPACE billentyűt. CLR-rel törölhetjük a képet. Ha elkészültünk az alapgenerációval, indítsunk az I-vel. Ekkor kirajzolódik az új generáció, majd négy funkció közül választhatunk.

1. Ha megnyomjuk a SPACE-t, akkor kirajzolja a következő generációt.

2. Ha a V-t nyomjuk le, akkor visszatérünk BASIC-be.

3. U hatására a program újból kezdődik.

4. Ha S-t választjuk, akkor ismét a tervező üzemmódba kerülünk: megjelenik a kurzor.

A program legelőször a gépi kódú kirajzoló rutint olvassa be.

MOLNÁR LÁSZLÓ

```
10 FORI=8192TO8399:READA:POKEI,A:S=S+A
20 NEXT:IFS<>28497THENPRINT"HIBA":END
30 POKE53280,6:POKE53281,0:PRINT"■□"
40 PRINT"■GENERACIO:  0     ELO SEJTEK:"
50 X=19:Y=12:GOTO160
60 GETA$:IFA$=""THEN60
70 IFA$="I"THENGOSUB310:GOTO200:REM IND
80 X1=0:Y1=0:IFA$="□"THENY1=-1:REM FEL
90 IFA$="■"THENY1=1:REM LE
100 IFA$="■"THENX1=-1:REM BALRA
110 IFA$="■"THENX1=1:REM JOBBRA
120 IFA$="□"THENRUN:REM CLR
130 GOSUB310:X=X+X1:Y=Y+Y1
140 IFY>23ORY<1THENY=Y-Y1
150 IFX>38ORX<1THENX=X-X1
160 B=X+1024+Y*40:IFA$=" "THEN180
170 GOSUB300:GOTO60
180 POKEB,(113-(PEEK(B)AND127))OR128
190 GOTO60
200 SYS8192:EL=PEEK(2)+PEEK(142)*256
210 G=G+1:PRINT"■"SPC(10)G"■";
220 PRINTSPC(27)"     ■■■■■"EL
230 IFEL=0THEN320
240 GETA$:IFA$=""THEN240
250 IFA$=" "THEN200:REM KOV. GENERACIO
260 IFA$="V"THENPRINT"□":END:REM VEGE
270 IFA$="U"THENRUN30:REM UJRA
280 IFA$="S"THEN50:REM MODOSITAS
290 GOTO240
300 POKEB,PEEK(B)OR128:RETURN
310 POKEB,PEEK(B)AND127:RETURN
320 PRINT"■"SPC(32)"KIHALT!"
330 FORI=0TO2000:NEXT:POKE198,0:RUN30
340 DATA 32,179,32,133,2,133,142
350 DATA 165,250,56,233,41,133,139
360 DATA 165,251,233,0,133,140,162
370 DATA 0,134,141,188,200,32,177
380 DATA 139,201,81,208,2,230,141
390 DATA 232,224,8,208,240,160,0
400 DATA 177,250,201,32,240,6,165
410 DATA 141,201,2,240,96,165,141
420 DATA 201,3,240,90,160,0,169
430 DATA 32,145,252,230,254,32,165
440 DATA 32,32,172,32,165,254,201
450 DATA 38,208,183,169,0,133,254
460 DATA 230,255,32,165,32,32,165
470 DATA 32,32,172,32,32,172,32
480 DATA 165,255,201,23,208,159,32
490 DATA 179,32,162,0,160,0,177
500 DATA 252,145,250,200,192,38,208
510 DATA 247,160,0,169,40,24,101
520 DATA 250,133,250,144,2,230,251
530 DATA 169,40,24,101,252,133,252
540 DATA 144,2,230,253,232,224,23
550 DATA 208,218,96,160,0,169,81
560 DATA 145,252,230,2,208,2,230
570 DATA 142,76,66,32,230,250,208
580 DATA 2,230,251,96,230,252,208
590 DATA 2,230,253,96,169,4,133
600 DATA 251,169,41,133,250,169,48
610 DATA 133,253,169,0,133,252,133
620 DATA 254,133,255,96,0,1,2
630 DATA 40,42,80,81,82
```

Örökélet C16-ra

Erre a géptípusra az egyik legismertebb játék a Paladine, de bosszantó, hogy a tizedik pályáig is eljuthatunk, mire elfogy az öt élet. Ilyenkor indíthatjuk elölről az egészet.

Kezdőknek ajánljuk az alábbiakat:

1. Töltsük be a programot LOAD-dal.

2. Írjuk be a POKE4234,234 POKE4235,234 POKE4236,234 parancsokat.

3. Indítsuk a játékot.

A program így örökéletes lesz, mert az életek számát csökkentő hármbájtos DEC utasítást 3 egybájtos NOP-pal cseréltük ki.

Még egy megjegyzés: a program a 15298-as ($2BC2) címen kezdődik.

IMRE ANDRÁS

Minden hétfőn 17-től 19 óráig

a Mikroszámítógép Magazin munkatársai és felkért szakértők válaszolnak az olvasók kérdéseire a szerkesztőségben: Budapest II., Fő u. 68. I. em. 109. vagy a 154-090 és a 154-250-es telefonon.

Minden kedden 17-től 20 óráig

ENTERPRISE-klub

a VSZM Közösségi Házban

(Bp. XI., Fehérvári út 120.)

Klubvezető: Romvári Gábor

Telefon: 450-860/473

Egy kis zene a Plus/4-en

Ha valaki próbált már komolyabban dallamokat kicsiholni a C16-os vagy Plus/4-es számítógépből, előbb-utóbb biztosan akadályokba ütközött. A SOUND utasítás

```
100 INPUT "KEZDOCIM";A
110 K1=A
120 READ B:POKE A,B:A=A+1
130 PRINT"■■ ELSO SZOLAM FELIRASA"
140 READ A$,B,C
150 POKE A,DEC(RIGHT$(A$,2)):POKE A+1,B:POKE A+2,C:A=A+3
160 IF B<>0 AND C<>0 THEN 140
170 REM ------------------------------
180 K2=A:READ B:POKEA,B:A=A+1
190 PRINT"■■ MASODIK SZOLAM FELIRASA"
200 READ A$,B,C
210 POKE A,DEC(RIGHT$(A$,2)):POKE A+1,B:POKE A+2,C:A=A+3
220 IF B<>0 OR C<>0 THEN 200
230 SYS 16416,K1,K2
240 REM
250 REM
260 REM ***************************
270 REM *                         *
280 REM * ITT KOVETKEZNEK A ZENE  *
290 REM *                         *
300 REM *        ADATAI           *
310 REM *                         *
320 REM ***************************
330 REM
340 DATA 3 :REM  *** ELSO SZOLAM ***
350 REM
360 DATA $60,50,10,$2A,50,10
370 DATA $60,50,10,$2A,50,10
380 DATA $60,25,5,$60,25,5
390 DATA $71,25,5,$81,25,5
400 DATA $60,25,5,$81,25,5
410 DATA $71,25,5,$2A,25,5
420 DATA $60,25,5,$60,25,5
430 DATA $71,25,5,$81,25,5
440 DATA $60,50,10,$2A,50,10
450 DATA $60,25,5,$60,25,5
460 DATA $71,25,5,$81,25,5
470 DATA $88,25,5,$81,25,5
480 DATA $71,25,5,$60,25,5
490 DATA $56,25,5,$2A,25,5
500 DATA $42,25,5,$56,25,5
510 DATA $60,50,10,$60,50,10
520 DATA $42,25,5,$2A,25,5
530 DATA $42,25,5,$2A,25,5
540 DATA $42,25,5,$56,25,5,$60,40,20
550 DATA $2A,25,5,$42,25,5
560 DATA $2A,25,5,$10,25,5
570 DATA $02,50,10,$2A,50,10
580 DATA $42,25,5,$2A,25,5
590 DATA $42,25,5,$2A,25,5
600 DATA $42,25,5,$56,25,5
610 DATA $60,25,5,$42,25,5
620 DATA $2A,25,5,$60,25,5
630 DATA $56,25,5,$71,25,5
640 DATA $60,50,10,$60,50,10
650 DATA $60,0,0
660 REM
670 REM-------------------------------
680 REM
690 DATA 2:REM *** MASODIK SZOLAM ***
700 REM
710 DATA $55,50,10,$C0,50,10
720 DATA $55,50,10,$C0,50,10
730 DATA $C0,50,10,$55,50,10
740 DATA $C0,50,10,$55,50,10
750 DATA $C0,50,10,$55,50,10
760 DATA $C0,50,10,$55,50,10
770 DATA $C0,50,10,$55,50,10
780 DATA $20,50,10,$E3,50,10
790 DATA $AD,50,10,$55,50,10
800 DATA $C0,50,10,$C0,50,10
810 DATA $83,50,10,$83,50,10
820 DATA $83,50,10,$83,50,10
830 DATA $55,50,10,$55,50,10
840 DATA $04,50,10,$55,50,10
850 DATA $83,50,10,$83,50,10
860 DATA $83,50,10,$83,50,10
870 DATA $55,50,10,$E3,50,10
880 DATA $C0,50,10,$C0,50,10
890 DATA $C0,0,0
```

1. lista

2. lista

```
100 rem ********************************
110 rem *                              *
120 rem * Ez a dallam  minden  olyan   *
130 rem *                              *
140 rem * esetben  jó, amikor valami   *
150 rem *                              *
160 rem * vagy valaki meghal.          *
170 rem *                              *
180 rem ********************************
190 rem
200 ?chr$(147);
210 input a:rem Kezdőcíme a dallam
adatainaK
220 K1=a:restore
230 read b:poke a,b:a=a+1
240 print"Első szólam felírása."
250 read a$,b,c
260 poKe a,dec(right$(a$,2)):poKe
a+1,b:poKe a+2,c:a=a+3
270 if b<>0 or c<>0 then 120
280 K2=a:read b:poKea,b:a=a+1
290 print"MásodiK szólam felírása."
300 read a$,b,c
310 poKe a,dec(right$(a$,2)):poKe
a+1,b:poKe a+2,c:a=a+3
320 if b<>0 or c<>0 then 140
330 sys 16416,K1,K2
340 rem
350 rem
360 rem ***************************
370 rem *                         *
380 rem * Itt KövetKeznek a zene  *
390 rem *                         *
400 rem *         adatai          *
410 rem *                         *
420 rem ***************************
430 rem
440 data 2 :rem  *** Első szólam ***
450 rem
460 data $04,50,10,$04,50,10
470 data $04,10,4,$04,50,10
480 data $55,50,10,$3b,10,4
490 data $3b,50,10,$04,10,4
500 data $04,50,10,$04,10,4
510 data $04,80,1,$04,0,0
520 rem
530 rem-------------------------------
540 rem
550 data 2:rem *** MásodiK szólam ***
560 rem
570 data $55,50,10,$55,50,10
580 data $55,10,4,$55,50,10
590 data $ad,50,10,$83,10,4
600 data $83,50,10,$55,10,4
610 data $55,50,10,$55,10,4
620 data $55,80,1,$55,0,0
```

Frekvencia táblázat 1.

Hang	Freky.	Érték	Felső byte	Alsó byte
C1	131	169	00	a9
	139	217	00	d9
D1	147	262	01	06
	156	305	01	31
E1	165	345	01	58
F1	175	383	01	7f
	185	419	01	a3
G1	196	453	01	c5
	208	485	01	e5
A1	220	516	02	04
	233	544	02	20
H1	247	571	02	3b
C2	262	597	02	55
	277	621	02	6d
D2	294	643	02	83
	311	665	02	99
E2	330	685	02	ad
F2	349	704	02	c0
	370	722	02	d2
G2	392	739	02	e3
	415	755	02	f3
A2	440	770	03	02
	466	784	03	10
H2	494	798	03	1e
C3	523	810	03	2a
	554	822	03	36
D3	587	834	03	42
	622	844	03	4c
E3	659	854	03	56
F3	698	864	03	60
	740	873	03	69
G3	784	881	03	71
	831	889	03	79
A3	880	897	03	81
	932	904	03	88
H3	988	911	03	8f

Frekvencia táblázat 2.

Hang	Freky.	Érték	Felső byte	Alsó byte
C4	1047	917	03	95
	1109	923	03	9b
D4	1175	929	03	a1
	1245	934	03	a6
E4	1319	939	03	ab
F4	1397	944	03	b0
	1480	948	03	b4
G4	1568	953	03	b9
	1661	957	03	bd
A4	1760	960	03	c0
	1865	964	03	c4
H4	1976	967	03	c7
C5	2093	971	03	cb
	2217	974	03	ce
D5	2349	976	03	d0
	2489	979	03	d3
E5	2637	982	03	d6
F5	2794	984	03	d8
	2960	986	03	da
G5	3136	988	03	dc
	3322	990	03	de
A5	3520	992	03	e0
	3729	994	03	e2
H5	3951	996	03	e4
C6	4186	997	03	e5
	4435	999	03	e7
D6	4699	1000	03	e8
	4978	1002	03	ea
E6	5274	1003	03	eb
F6	5588	1004	03	ec
	5920	1005	03	ed
G6	6272	1006	03	ee
	6645	1007	03	ef

Kódtáblázatok

ugyanis csak egy hangot ad ki, azt is csak bizonyos ideig, így a bonyolultabb dallamok megalkotása hátráltatja a program futását. Nehézkes dolog megoldani egy dallam lejátszását, miközben netán mást is szeretnénk végrehajtani a géppel.
Az *1. listán* közölt program bizonyos mértékig könnyít ezen a problémán, bár profi zenék nem készülhetnek vele.

Ami viszont igen

A programmal könnyen állíthatunk elő kétszólamú muzsikát, amely a gépen futó BASIC programtól függetlenül szól. A memóriában tetszőleges helyen tárolhatók a dallamok adatai, akár egyszerre több dallamé is. Egy dallamot a SYS utasítással indíthatunk el úgy, hogy az utasításban különkülön megadjuk a két szólam kezdőcímét.

Egy hangot három bájttal írunk le: az első a hang értékének alsó bájtja, a második a hang hosszát, a harmadik pedig a hang utáni szünet hosszát határozza meg. A memóriában a szólam adatait egymás után kell megadni, kötelezően három adatot egy hanghoz. A szólam adatsorát három darab bájttal zárjuk le. Ugyanezt tesszük mindkét szólamnál külön-külön, majd a zenét egy

SYS 16416,Kcím1,Kcím2

utasítással indítjuk el. Ha ki akarjuk kapcsolni a muzsikát, még mielőtt magától leállna, a SYS 17000 utasítást írjuk be.

Hogyan működik a program?

Az ötvened másodpercenként generálódó rendszermegszakításra kapcsolódik rá a program. Minden egyes híváskor külön megnézi a két szólamot, vagyis hogy éppen hang szól-e, illetve szünet van-e. Mindkét esetben a megfelelő számlálók értékét csökkenti eggyel, és a szünetek után új hangot vesz be a tárból. Teszi ezt mindaddig, amíg a hanghossz és a szünet helyén nullát nem talál. Azt, hogy a program aktív-e, úgy tudjuk megállapítani, hogy megnézzük: a $0315-ös bájt (a rendszermegszakítás vektorának felső bájtja) mit tartalmaz. Ha ott $CE-t találunk, akkor a program nem aktív, vagyis a zene nem szól.

Ahhoz, hogy programunk megfelelően működjön, még egy nagyon fontos dolgot kell tudnunk. Aki ismeri a Plus/4 vagy a C16-os hanggenerálását, már biztosan észrevette, hogy a hang értékének csak az alsó bájtja adható meg. Ez pusztán azért van, mert így minden hanghoz csak három adatot kell közölnünk. Viszont a szólam adatainak elején meg kell adni az egész szólamra jellemző bájtot. De ez nem sokkal nehezíti meg a dolgunkat, mert a 02-es értékhez egy oktáv, a 03-as értékhez pedig négy oktáv tartozik.

Azok számára, akiket esetleg nem érdekel a szakmai oldal, készítettem a *2. listán* látható BASIC programot, amely egy kész dallamot tartalmaz. A program betöltés és futtatás után megkérdezi az adatok kezdőcímét (ide 20000-et érdemes írni), majd betölti az adatokat és indítja a zenét.

Ha pedig valaki saját dallamokat akarna programozni, de nincs kódtáblázata, amiből a hangok adatait kikereshetné, használja a saját készítésű kódtáblázataimat.

KORSÓS ISTVÁN

Egy apró, de érdekes program C64-re

Aki rászán néhány percet az alábbi programra, meglepő élményben részesül. A képernyő csíkossá válik, ám nem vízszintesen, hanem függőlegesen. A csíkokat a RESTORE gomb lenyomásával mozgatni is lehet. Igaz, hogy a programnak a vizuális hatáson kívül semmi gyakorlati haszna nincs, de betehetjük egy-egy saját készítésű képújság végére, esetleg adásszünet idejére.

BÁRTFAI BARNABÁS

```
10  REM * BB SOFTWARE   1987. *
20  FOR I=0 TO 58
30  READ A:POKE 16378+I,A
40  S=S+A:NEXT
50  IF S<>6373 THEN STOP
60  SYS 16378
70  DATA 169,11,141,17,208
80  DATA 120,169,0,141,32,208,169,9,141
90  DATA 32,208,169,6,141,32,208,169,2
100 DATA 141,32,208,169,8,141,32,208,169
110 DATA 14,141,32,208,169,5,141,32,208
120 DATA 169,15,141,32,208,169,7,141,32
130 DATA 208,169,1,141,32,208,76,0,64
```

SZÁMÍTÁSTECHNIKAI ÉS ROKON SZAKMÁK

Továbbtanulás közép- és felső fokon

Lapunk 1984. évi 6. számában beszámoltunk arról, hogy a közép- és felsőfokú tanintézetekben milyen lehetőségek voltak általános és speciális számítástechnikai ismeretek elsajátítására. Azóta jelentősen szélesedtek az ilyen irányú továbbtanulási lehetőségek, s javultak a képzés feltételei is. A pályaválasztás megkönnyítése céljából közöljük az ez évi aktualizált összeállítást az iskolarendszerű középfokú és az intézményes felsőfokú elektronikai, informatikai, számítástechnikai és szervezési jellegű továbbtanulási lehetőségekről.

Szakmunkásképző iskolák

Elektronikai műszerész

14. Sz. Ifjú Gárda Ipari Szakmunkásképző Intézet, Budapest
609. Sz. Türr István Ipari Szakmunkásképző Intézet, Baja
30. Sz. Ságvári Endre Ipari Szakmunkásképző Intézet, Budapest
402. Sz. Hunyadi Mátyás Ipari Szakmunkásképző Intézet, Mosonmagyaróvár
403. Sz. Pesti Barnabás Ipari Szakmunkásképző Intézet, Sopron
621. Sz. Ipari Szakmunkásképző Intézet, Kiskőrös
611. Sz. Alfredo Lima Ipari Szakmunkásképző Intézet, Békéscsaba
101. Sz. Ipari Szakmunkásképző Intézet, Miskolc
601. Sz. József Attila Ipari Szakmunkásképző Intézet, Makó
320. Sz. Árpád Ipari Szakmunkásképző Intézet, Székesfehérvár
306. Sz. Táncsics Mihály Ipari Szakmunkásképző Intézet, Veszprém
203. Sz. Bem József Ipari Szakmunkásképző Intézet, Cegléd
213. Sz. Ipari Szakmunkásképző Intézet, Hatvan
317. Sz. Ipari Szakmunkásképző Intézet, Esztergom
610. Sz. Sághy Mihály Ipari Szakmunkásképző Intézet, Csongrád
401. Sz. Kossuth Lajos Ipari Szakmunkásképző Intézet, Győr
521. Sz. Ipari Szakmunkásképző Intézet, Fonyód
211. Sz. Ipari Szakmunkásképző Intézet, Salgótarján
217. Sz. Szondi György Ipari Szakmunkásképző Intézet, Balassagyarmat
503. Sz. Ipari Szakmunkásképző Intézet, Kaposvár
529. Sz. Ipari Szakmunkásképző Intézet, Tab
526. Sz. Farkas J. Ipari Szakmunkásképző Intézet, Csurgó
107. Sz. Mező Imre Ipari Szakmunkásképző Intézet, Nyíregyháza
619. Sz. Ipari Szakmunkásképző Intézet, Kalocsa
606. Sz. Ipari Szakmunkásképző Intézet, Jászberény
508. Sz. Tarr Imre Ipari Szakmunkásképző Intézet, Pécs
406. Sz. Polai János Ipari Szakmunkásképző Intézet, Nagykanizsa
407. Sz. Munkácsy Mihály Ipari Szakmunkásképző Intézet, Zalaegerszeg
223. Sz. Ipari Szakmunkásképző Intézet, Nagykáta
224. Sz. Ipari Szakmunkásképző Intézet, Nagykőrös

Irodagép-műszerész

14. Sz. Ifjú Gárda Ipari Szakmunkásképző Intézet, Budapest
618. Sz. Ipari Szakmunkásképző Intézet, Kiskunfélegyháza
401. Sz. Kossuth Lajos Ipari Szakmunkásképző Intézet, Győr
623. Sz. Vágó Béla Ipari Szakmunkásképző Intézet, Kecskemét
624. Sz. Ipari Szakmunkásképző Intézet, Szeged
101. Sz. Ipari Szakmunkásképző Intézet, Miskolc
311. Sz. Ipari Szakmunkásképző Intézet, Tata
127. Sz. Ipari Szakmunkásképző Intézet, Debrecen
508. Sz. Tarr Imre Ipari Szakmunkásképző Intézet, Pécs
211. Sz. Ipari Szakmunkásképző Intézet, Salgótarján
212. Sz. Ipari Szakmunkásképző Intézet, Eger
217. Sz. Szondi György Ipari Szakmunkásképző Intézet, Balassagyarmat
224. Sz. Ipari Szakmunkásképző Intézet, Nagykőrös
304. Sz. Acsadi Ignác Ipari Szakmunkásképző Intézet, Pápa
320. Sz. Árpád Ipari Szakmunkásképző Intézet, Székesfehérvár
503. Sz. Rippl-Rónai J. Ipari Szakmunkásképző Intézet, Kaposvár
306. Sz. Táncsics Mihály Ipari Szakmunkásképző Intézet, Veszprém
405. Sz. Berzsenyi Dániel Ipari Szakmunkásképző Intézet, Szombathely
407. Sz. Munkácsy Mihály Ipari Szakmunkásképző Intézet, Zalaegerszeg

Számítástechnikai műszerész

316. Sz. Makarenkó Ipari Szakmunkásképző Intézet, Dunaújváros

Irányítástechnikai műszerész

14. Sz. Ifjú Gárda Ipari Szakmunkásképző Intézet, Budapest
612. Sz. Ipari Szakmunkásképző Intézet, Orosháza
105. Sz. Lékai János Ipari Szakmunkásképző Intézet, Kazincbarcika
505. Sz. Ipari Szakmunkásképző Intézet, Szekszárd
316. Sz. Makarenkó Ipari Szakmunkásképző Intézet, Dunaújváros
318. Sz. Ipari Szakmunkásképző Intézet, Komárom

Elektroműszerész

1. Sz. Bajáki Ferenc Ipari Szakmunkásképző Intézet, Budapest
14. Sz. Ifjú Gárda Ipari Szakmunkásképző Intézet, Budapest
22. Sz. Puskás Tivadar Ipari Szakmunkásképző Intézet, Budapest
30. Sz. Ságvári Endre Ipari Szakmunkásképző Intézet, Budapest
618. Sz. Ipari Szakmunkásképző Intézet, Kiskunhalas
621. Sz. Ipari Szakmunkásképző Intézet, Kiskőrös
104. Sz. Debreceni M. Ipari Szakmunkásképző Intézet, Miskolc
601. Sz. József Attila Ipari Szakmunkásképző Intézet, Makó
624. Sz. Ipari Szakmunkásképző Intézet, Szeged
401. Sz. Kossuth Lajos Ipari Szakmunkásképző Intézet, Győr
220. Sz. Ipari Szakmunkásképző Intézet, Érd
127. Sz. Ipari Szakmunkásképző Intézet, Debrecen
107. Sz. Mező Imre Ipari Szakmunkásképző Intézet, Nyíregyháza
203. Sz. Bem József Ipari Szakmunkásképző Intézet, Cegléd
201. Sz. Gárdonyi Géza Ipari Szakmunkásképző Intézet, Dunakeszi
223. Sz. Ipari Szakmunkásképző Intézet, Nagykáta
122. Sz. Ipari Szakmunkásképző Intézet, Hajdúböszörmény
222. Sz. Ipari Szakmunkásképző Intézet, Kiskunlacháza
629. Sz. Ragd Antal Ipari Szakmunkásképző Intézet, Karcag
111. Sz. II. Rákóczi Ferenc Ipari Szakmunkásképző Intézet, Kisvárda
405. Sz. Berzsenyi Dániel Ipari Szakmunkásképző Intézet, Szombathely
604. Sz. Ipari Szakmunkásképző Intézet, Törökszentmiklós
300. Sz. Ipari Szakmunkásképző Intézet, Ajka

Távközlés-technikai műszerész

14. Sz. Ifjú Gárda Ipari Szakmunkásképző Intézet, Budapest
306. Sz. Táncsics Mihály Ipari Szakmunkásképző Intézet, Veszprém
618. Sz. Ipari Szakmunkásképző Intézet, Kiskunhalas
608. Sz. Ipari Szakmunkásképző Intézet, Kiskunfélegyháza
609. Sz. Türr István Ipari Szakmunkásképző Intézet, Baja
127. Sz. Ipari Szakmunkásképző Intézet, Debrecen

Szakmunkásképzést folytató ipari szakközépiskolák

Elektronikai műszerész

Kolos Richárd Ipari Szakközépiskola, Budapest
Bolyai János Híradástechnikai Szakközépiskola, Budapest
Egressy Gábor Finommechanikai és Műszeripari Szakközépiskola, Budapest
Corvin Mátyás Híradástechnikai Szakközépiskola, Budapest
Bajáki Ferenc Szakközépiskola és Szakmunkásképző Intézet, Budapest
Rudnay Gyula Ipari Szakmunkásképző Intézet és Szakközépiskola, Tab
Kossuth Zsuzsa Finommechanikai, Műszeripari és Gépészeti Szakközépiskola, Hódmezővásárhely
Lőwy Sándor Szakközépiskola, Vác
Ipari Szakközépiskola, Kaposvár
Bottyán János Finommechanikai és Műszeripari Szakközépiskola, Esztergom

Irodagép-műszerész

Finommechanikai és Műszeripari Szakközépiskola, Budapest

Számítástechnikai műszerész

Landler Jenő Ipari Szakközépiskola, Budapest
Ságvári Endre Szakközépiskola, Székesfehérvár
623. Sz. Vágó Béla Ipari Szakmunkásképző Intézet és Szakközépiskola, Kecskemét
Latinca Sándor Gép- és Villamosipari Szakközépiskola, Budapest

Elektroműszerész

Bajáki F. Ipari Szakközépiskola és Szakmunkásképző Intézet, Budapest
Ipari Szakközépiskola és Szakmunkásképző Intézet, Ajka
Bolyai János Híradástechnikai Szakközépiskola, Budapest
Bagi Ilona Ipari Szakközépiskola, Budapest
Latinca Sándor Gép- és Villamosipari Szakközépiskola, Budapest
Zipernovszky K. Szakközépiskola, Pécs
508. Sz. Ipari Szakmunkásképző Intézet és Szakközépiskola, Pécs
623. Sz. Vágó Béla Ipari Szakmunkásképző Intézet és Szakközépiskola, Kecskemét
Kossuth Zsuzsa Szakközépiskola, Hódmezővásárhely
Landler Jenő Ipari Szakközépiskola, Debrecen
Ságvári Endre Ipari Szakközépiskola, Székesfehérvár
611. Sz. Alfredo Lima Ipari Szakmunkásképző Intézet és Szakközépiskola, Békéscsaba
Gép- és Műszeripari Szakközépiskola, Eger
Vak Bottyán János Műszeripari és Gépészeti Szakközépiskola, Gyöngyös
Bottyán János Finommechanikai és Műszeripari Szakközépiskola, Esztergom
211. Sz. Ipari Szakmunkásképző Intézet és Szakközépiskola, Salgótarján
Bródi Imre Ipari Szakközépiskola, Ózd
2. Sz. Ipari Szakközépiskola, Miskolc
405. Sz. Berzsenyi Dániel Ipari Szakmunkásképző Intézet és Szakközépiskola, Szombathely

Irányítástechnikai műszerész

Egressy Gábor Finommechanikai és Műszeripari Szakközépiskola, Budapest
106. Sz. Ipari Szakmunkásképző Intézet és Szakközépiskola, Leninváros
Gép- és Műszeripari Szakközépiskola, Eger
100. Sz. Ipari Szakmunkásképző Intézet és Szakközépiskola, Miskolc
105. Sz. Ipari Szakmunkásképző Intézet és Szakközépiskola, Kazincbarcika
Pálfy János Műszeripari és Vegyipari Szakközépiskola, Szolnok
Vak Bottyán János Ipari Szakközépiskola, Gyöngyös
Irányítástechnikai Szakmunkásképző Intézet és Szakközépiskola, Füzfőgyártelep

Távközléstechnikai műszerész

Puskás Tivadar Híradástechnikai Szakközépiskola, Budapest
Széchenyi István Gimnázium és Szakközépiskola, Pécs
Bebrits Lajos Szakközépiskola, Szeged
Mechwart András Gépipari Szakközépiskola, Debrecen
Landler Jenő Ipari Szakközépiskola, Debrecen
I. István Gépgyártástechnológiai és Híradástechnikai Szakközépiskola, Esztergom
Híradásipari Szakmunkásképző Intézet és Szakközépiskola, Fonyód
Pataky István Híradásipari Szakközépiskola, Budapest
Bánki Donát Ipari Szakközépiskola, Nyíregyháza
2. Sz. Ipari Szakközépiskola, Miskolc
403. Sz. Pesti Barnabás Ipari Szakmunkásképző Intézet és Szakközépiskola, Sopron

Gimnáziumok

A matematikai törzsanyag keretében minden tanuló elsajátítja a tantárgyhoz kapcsolódó számítástechnikai alapismereteket, a kiegészítő anyagok pedig az adott témakörökhöz illesztett számítógépes felhasználáshoz mutatnak be célszerű feladatmegoldási lehetőségeket is.

A számítástechnikai alapismereteket még magasabb óraszámban tanítják a következő iskolák speciális tantervű matematikai osztályaiban:

Széchenyivárosi Gimnázium, Kecskemét
Földes Ferenc Gimnázium, Miskolc
JATE Ságvári Endre Gyakorló Gimnázium, Szeged
Teleki Blanka Gimnázium, Székesfehérvár
Révai Miklós Gimnázium, Győr
Fazekas Mihály Gimnázium, Debrecen
Táncsics Mihály Gimnázium, Kaposvár
Krúdy Gyula Gimnázium, Nyíregyháza
Verseghy Ferenc Gimnázium, Szolnok
Lovassy László Gimnázium és Szakközépiskola, Veszprém
Zrínyi Miklós Gimnázium, Zalaegerszeg
Fazekas Mihály Fővárosi Gyakorló Gimnázium, Budapest
Berzsenyi Dániel Gimnázium, Budapest
I. István Gimnázium, Budapest

Az I—IV. évfolyamon egyaránt helyileg meghatározott 3 órában oktatott technika tantárgy — iskolánként választható módon — a következő változatok szerint nyújt számítástechnikai és azzal rokon alapismereteket:

„A" változat: automatizálás, ipari elektronika, számítástechnika,
„B" változat: irányítástechnika,
„C" változat: elektronika,
„D" változat: informatika.

A fakultatív tantárgyi oktatás keretein belül — iskolánként választható módon — lehetőség van:

— 4 éven át 310 óra ráfordítással számítógép-kezelői (operátori) szakképesítés megszerzésére,
— 4 éven át 310 óra ráfordítással helyileg kialakított tanterv szerinti számítástechnikai oktatásban való részvételre,
— 4 éven keresztül évi 72 óra ráfordítással, előírt tantervű programozási és egyéb számítástechnikai alapismeretek megszerzésére.

Eredményes záróvizsga esetén szakképesítést nyújtó, tehát előírt munkakörök betöltésére jogosító számítógép-kezelői (operátori) képzés az 1987/88. tanévben a következő iskolákban folyik:

Nagy Lajos Gimnázium, Pécs
Rózsa Ferenc Gimnázium, Békéscsaba
Kiss Lajos Gimnázium és Szakközépiskola, Gyomaendrőd
Erkel Ferenc Gimnázium és Szakközépiskola, Gyula
Táncsics Mihály Gimnázium és Szakközépiskola, Orosháza
Péter András Gimnázium és Szakközépiskola, Szeghalom
Ságvári Endre Gimnázium, Kazincbarcika
Földes Ferenc Gimnázium, Miskolc
Zrínyi Ilona Gimnázium, Miskolc
József Attila Gimnázium és Szakközépiskola, Ózd
Radnóti Miklós Gimnázium, Szeged
Tömörkény István Gimnázium és Szakközépiskola, Szeged
Teleki Blanka Gimnázium, Székesfehérvár
Bocskai István Gimnázium, Biharkeresztes
Fazekas Mihály Gimnázium, Debrecen
Ady Endre Gimnázium, Debrecen
Tóthfalusi Sándor Gimnázium, Derecske
Gimnázium és Szakközépiskola, Füzesabony
Ráday Pál Gimnázium, Pécel
Móricz Zsigmond Gimnázium, Szentendre
Petőfi Sándor Gimnázium és Szakközépiskola, Aszód
Baktay Ervin Gimnázium és Szakközépiskola, Dunaharaszti
Gimnázium, Barcs
Munkácsy Mihály Gimnázium és Szakközépiskola, Kaposvár
Csokonai Vitéz Mihály Gimnázium, Csurgó
Kossuth Lajos Gimnázium, Nyíregyháza
Zalka Máté Gimnázium, Fehérgyarmat
Teleki Blanka Gimnázium, Tiszalök
József Attila Gimnázium, Kunszentmárton
Petőfi Sándor Gimnázium, Bonyhád
Béri Balogh Ádám Gimnázium, Tamási
Garay János Gimnázium és Szakközépiskola, Szekszárd
Bródy Imre Gimnázium, Ajka
Ságvári Endre Gimnázium, Zalaegerszeg
Dr. Mező Ferenc Gimnázium és Szakközépiskola, Nagykanizsa
III. ker. Árpád Gimnázium, Budapest
Könyves Kálmán Gimnázium, Budapest
Zrínyi Miklós Gimnázium, Budapest
Táncsics Mihály Gimnázium, Budapest
Kilián György Gimnázium, Budapest
Fürst Sándor Gimnázium, Budapest
Nagy László Általános Iskola és Gimnázium, Budapest
Jedlik Ányos Gimnázium, Budapest

Kevés kivételtől eltekintve, valamennyi gimnáziumban működnek számítástechnikai diákkörök (szakkörök) kezdők és haladók számára, általában heti 2 órás foglalkozásokkal.

Számítástechnikai programozó

Hámán Kató Közgazdasági Szakközépiskola, Budapest (II. ker., Jurányi u.)
Közgazdasági Szakközépiskola, Budapest (XIV. ker., Kerepesi út)
Csány László Közgazdasági Szakközépiskola, Zalaegerszeg
Gimnázium és Közgazdasági Szakközépiskola, Eger
Sebes György Közgazdasági Szakközépiskola, Békéscsaba
Kada Elek Közgazdasági Szakközépiskola, Kecskemét
Kőrösy J. Közgazdasági és Kereskedelmi Szakközépiskola, Szeged
Közgazdasági Szakközépiskola, Százhalombatta
Paksi Atomerőmű Műszaki Szakközépiskola, Paks

Számítástechnikai folyamatszervező

Hámán Kató Közgazdasági Szakközépiskola, Budapest
Csány László Közgazdasági Szakközépiskola, Zalaegerszeg
Gimnázium és Közgazdasági Szakközépiskola, Eger
Sebes György Közgazdasági Szakközépiskola, Békéscsaba
Noszlopy Gáspár Közgazdasági Szakközépiskola, Kaposvár
Fáy András Közgazdasági Szakközépiskola, Miskolc
Vásárhelyi Pál Közgazdasági Szakközépiskola, Szolnok
Kada Elek Közgazdasági Szakközépiskola, Kecskemét
Bethlen Gábor Közgazdasági Szakközépiskola, Debrecen
Közgazdasági Szakközépiskola, Százhalombatta
Lengyel Gyula Közgazdasági Szakközépiskola, Győr
Rudas László Közgazdasági Szakközépiskola, Dunaújváros
Nádasdy Tamás Közgazdasági Szakközépiskola, Csepreg

Középfokú számítástechnikai szakmai alkalmazói képzés

Középfokú számítástechnikai szakmai alkalmazói képzést folytatnak a számviteli-gazdálkodási és pénzügyi ágazatú közgazdasági szakközépiskolák és az olyan műszaki szakközépiskolák, amelyekben a képzés technikus-, illetve szakmunkásképzésre ágazik el. Folyamatosan vezetik be a számítástechnikai alkalmazói képzést az egyéb szakközépiskolákban is.

Felsőfokú oktatási intézmények

Az *Eötvös Loránd Tudományegyetem* természettudományi karán (Budapest) és a *József Attila Tudományegyetem* természettudományi karán (Szeged) programozó matematikusok és programtervező matematikusok képzése folyik. Ezenkívül a budapesti *ELTE-TTK* középiskolai kétszakos tanári képzésének keretében matematika—számítástechnika, a szegedi *JATE* nem tanári természettudományi szakán pedig közgazdasági programtervező matematikusok kiképzése történik. A programozó és a programtervező matematikus, valamint a közgazdasági programtervező matematikus szakok kétlépcsősek. A 6 féléves első lépcső közösen indul és főiskolai szintű oklevelet adó programozó matematikusi végzettség megszerzésével zárul. Ezt követően a hallgatók újbóli felvétel során kerülhetnek a programtervező matematikusi, illetve a közgazdasági programtervező matematikusi szakokra, amelyeken 4 félév tanulmányi idő után egyetemi szintű oklevelet nyerhetnek.

A *Kossuth Lajos Tudományegyetem* természettudományi karán (Debrecen) matematika—ábrázoló geometria és számítástechnika szakos középiskolai tanárokat, valamint programozó matematikusokat képeznek. A matematika—fizika, kémia—fizika, fizika—technika tanár szakos hallgatók harmadévtől felvehetik a számítástechnika szakot. A programozó matematikus szak elvégzése után — a legkiválóbbak — az *ELTE-TTK-n* programtervező matematikus szakot végezhetnek.

A *Marx Károly Közgazdaságtudományi Egyetem* (Budapest) általános közgazdaság karának közgazdásztanári szakán információelméleti és -feldolgozási szakos tanárokat képeznek a közgazdasági szakközépiskolák számára. A tervgazdasági szak országos és területi irányító szervek, tudományos intézetek, nagyvállalatok részére képez — a más közgazdasági szakokhoz képest elmélyültebb elméleti, matematikai és számítástechnikai ismeretekkel rendelkező — közgazdászokat. Az egyetemen működő többi szak a népgazdaság különböző ágazatai számára képez vállalati és népgazdasági szintű gazdasági elemzést, tervezést, szervezést, fejlesztést és irányítást ellátni képes közgazdászokat.

A *Janus Pannonius Tudományegyetem* közgazdaságtudományi karán (Pécs) az ipari tervező-szervező, a mezőgazdasági és az áruforgalmi szakon vállalati és népgazdasági szintű gazdasági elemzést, tervezést, fejlesztést és irányítást ellátni képes közgazdászokat képeznek.

A *Budapesti Műszaki Egyetem* villamosmérnöki karán elektronikai, számítástechnikai jellegű szakképzés folyik a híradástechnikai, a műszer- és irányítástechnikai, a mikroelektronikai és technológiai, s az informatika szakon. Ilyen jellegű képzést folytatnak a közlekedésmérnöki kar közlekedési rendszerszervező ágazatán is.

A *Nehézipari Műszaki Egyetem* (Miskolc) gépészmérnöki karának termelési rendszer szakán a természettudományi, konstrukciós és technológiai ismeretekkel rendelkező gépészmérnökök üzemszervezési irányú szakosítása történik. A géptervező szak folyamattervező ágazatán is nagy súllyal szerepelnek a különböző számítástechnikai témakörök, numerikus módszerek, operációkutatás stb. A Kohó- és Fémipari Főiskolai Kar (Dunaújváros) szervezési szakán számítástechnikai, középiskolai műszaki tanári szakán pedig informatikai szakirány működik. A Közgazdaságtudományi Intézet (Miskolc) vállalatgazdálkodási szakán fontos szerepet tölt be az informatikai oktatás.

A *Veszprémi Vegyipari Egyetem* szervező vegyészmérnöki, szervező vegyész üzemmérnöki, valamint vegyipari műszer- és méréstechnikai szakán folyik magasabb szintű szervezési, rendszerelméleti, rendszertechnikai, illetve számítástechnikai ismeretek oktatása.

A mezőgazdaság-tudományi és társadalomtudományi karral működő *Gödöllői Agrártudományi Egyetemen* üzemszervező agrármérnökök és üzemszervező üzemmérnökök képzése folyik.

A külkereskedelmi áruforgalmi és külkereskedelmi idegen nyelvű levelező szakkal működő *Külkereskedelmi Főiskola* (Budapest) szakosított továbbképzésének keretében informatikai képzés folyik. A szakosított továbbképzésre való jelentkezéshez nyelvvizsga szükséges.

A *Pénzügyi és Számviteli Főiskolán* (Budapest, Salgótarján) olyan adatfeldolgozási rendszerszervezők képzése folyik, akik személyi számítógép- és számítástechnikai, hardver-, szoftverismeretekkel, szervezéstechnológiai eljárásokkal képesek a vállalatok és intézmények modellezésére, informatikai szabályozására. Az elsődleges képzési célnak megfelelően a hallgatók elsajátítják a legfontosabb számítástechnikai, informatikai, vezetési módszertani ismereteket. Emellett megfelelő pénzügyi, számviteli, elemzési stb. ismeretekre is szert tesznek.

A *Budapesti Bánki Donát Gépipari Műszaki Főiskola* gép- és rendszertechnikai, valamint szervező és informatikai szakot működtet. A levelező tagozatú miskolci konzultációs központban is folyik szervező- és informatikai képzés, szervezői ágazat pedig a felsőfokú végzettségűek budapesti szaküzemmérnöki képzésében is van.

A *Gépipari és Automatizálási Műszaki Főiskola* (Kecskemét) gépipari automatizálási szakán automatizálási, valamint számítógéptechnikai ágazat működik.

A *Kandó Kálmán Villamosipari Műszaki Főiskola* (Budapest) mikroelektronika, alkatrész- és készüléktechnológiai szakán félvezető- és mikroelektronikai technológia, elektroncső- és fényforrás-technológia, valamint elektronikai készüléktechnológia szakágak működnek. A műszeripari és automatizálási szak szakágai: folyamatszabályozás, digitális irányítástechnika, elektronikus műszerek, orvostechnika. A híradásipari szak számítástechnikával összefüggő szakágai: átviteltechnika és adatátvitel, kapcsolástechnika, mikrohullámú technika. Az informatika szakon szoros értelemben vett számítástechnika-alkalmazást sajátítanak el a hallgatók. A számítástechnikai eszközök szakon (Székesfehérvár) a számítástechnikai eszközök gyártására, üzemeltetésére készítik fel a hallgatókat.

A *Pollack Mihály Műszaki Főiskola* (Pécs) műszaki informatika szakot működtet.

A *Győri Széchenyi István Közlekedési és Távközlési Műszaki Főiskola* vezetékes távközlés-technikai, vezeték nélküli távközlés-technikai és közlekedési automatikai szakjain folyó képzés kapcsolódik szorosan az elektronikai diszciplinákhoz.

Az *Ybl Miklós Építőipari Műszaki Főiskola* (Budapest, Debrecen) felsőfokú végzettségűek számára szolgáló szaküzemmérnöki szakjain gazdasági-szervezői képzés folyik.

A felsoroltakon kívül a tanárképző egyetemeken és főiskolákon kellően felkészítik a leendő tanárokat (köztük a matematika, technika szakosokat, a kereskedelmi és vendéglátó-ipari gazdasági szaktanárokat, szakoktatókat stb.) a szükséges számítástechnikai alkalmazási feladatok ellátására. Az egyetemeken és főiskolákon — a jelzett karokon, szakokon és ágazatokon folyó speciális számítástechnikai képzésen kívül — általában az egyéb hallgatók is kapnak számítástechnikai szakmai alkalmazási felkészítést.

Dr. Kóbor Zoltán

BASIC és gépi kód

Legutóbb az USR függvényről írtam. Említettem, hogy az IBM PC BASIC-je módot nyújt arra, hogy egyidejűleg tíz USR függvény közül bármelyik hívható legyen, és ezek belépési pontjait a POKE utasításpárok helyett DEF USR utasítással határozzuk meg. Hasonlóra a Commodore gépeken is van lehetőség. Most néhány szót ejtek egy átmeneti megoldásról, majd — előkészítendő a végleges változatot — ismertetem a beszúrás fogalmát.

Két POKE helyett egy SYS

Egy kicsivel közelebb visz a megoldáshoz, ha készítünk egy gépi kódú szubrutint, amelyet SYS cím1,cím2 szintaktikájú utasítással hívhatunk meg. Itt cím1 a SYS-szel hívott rutinnak, cím2 pedig az aktivizálandó USR függvénynek a belépési címe. Mindkettő tetszőleges aritmetikai kifejezés lehet. Célszerű változókat használni, és értéküket a BASIC program elején beállítani.

Az eljárás annak a módszernek a kiterjesztése, amelyet Németh Béla használt az 1987/7. számban közölt programjaiban. Ha egy programból csak egy USR függvényt használunk, nem érdemes bonyolítani a dolgot: jobb az említett programokban alkalmazott módszernél maradni.

A gépi kódú szubrutin szimbolikus assembly kódja a *listán* látható. Az egyes géptípusokhoz tartozó címeket a *táblázatból* olvashatjuk ki; ezek mindegyikéről írtam már. A rutin a RAM bármely szabad területén elhelyezhető, ezért hagytam el a sorok elejéről az utasítások címét. Aki érdemesnek találja, próbálja meg a rutint — akár mauálisan, akár assemblerrel — lefordítva bevinni a gépébe. A C16-os változat a ROM futinokra vonatkozó korábbi információk birtokában lényegesen rövidebbé tehető.

```
jsr chkcom
jsr frmnum
jsr getadr
lda linnum
ldy linnum+1
sta usrvec
sty usrvec+1
rts
```

Programlista

A beszúrásokról

A gépi kódú rutinok BASIC-ből való aktivizálásának a SYS és USR mellett igen elterjedt módszere az ún. beszúrás, ismert angol szóval wedge. Lényege, hogy a BASIC interpreter vagy a KERNAL rutinjait „eltérítjük", saját rutinjainkra irányítjuk. Így működnek a különböző BASIC-bővítések, segédprogramok. Jellemző példák a beszúrásra a floppymeghajtókhoz adott test/demo-lemezen található VIC 20 WEDGE és C64 WEDGE.

A beszúrásra több módszert ismerünk. Legnépszerűbb a BASIC és KERNAL vektorok átírása, amiről alább egy kicsit részletesebben írok. Közismert még a CHRGET rutin eltérítése, melynek sok félresikerült alkalmazásával találkoztam már. Speciálisabb az a — VC20-on és tárbővítés nélküli C16-on nem alkalmazható — eljárás, melynek lényege, hogy a ROM tartalmát a vele azonos címen levő RAM-ba másoljuk, ott végrehajtjuk a szükséges módosításokat, majd az adatirány-regiszter beállításával a RAM-ot „tesszük láthatóvá" a processzor számára.

név	VC20	C64	C16
chkcom	$cefd	$aefd	$9491
frmnum	$cd8a	$ad8a	$9314
getadr	$d7f7	$b7f7	$9de4
linnum	$14	$14	$14
usrvec	$01	$0311	$0501

A rendszer-címek táblázata

A vektorok átírásának speciális esete a C16-on a felhasználói tokenek használata. Ezt a témát részletesen ismerteti Gnädig Péter a Mikroszámítógép Magazin 1987/4. számában.

A vektorok átírása

A hármas memórialap elején találhatjuk meg a rendszervektorok táblázatát. Ez a címtáblázat két részből áll: elöl a BASIC rutinok, mögöttük a KERNAL rutinok belépési pontjainak címei. A három géptípuson a táblázatok szerkezete némileg eltérő, de azok a címek, amelyek a mostani feladat szempontjából fontosak, mindhárom gépen azonos helyen találhatók meg.

A vektortáblázat a beépített rendszer tervszerű előkészítése a beszúrásokra. A ROM programok egyes pontjain olyan közvetett ugróutasításokat találunk, melyeknek operandusa valamelyik vektor. Az esetek többségében az ugrás az indirekt JMP utáni utasításra történik. Első pillantásra ez ostobaságnak látszik, pedig nem az: ez teszi lehetővé, hogy a ROM-ban lévő rutin futásába beavatkozhassunk.

Bennünket most két BASIC vektor érdekel. A DEF USR utasítás felismeréséhez és kezeléséhez a $0308 . . . $0309 címeken levő *utasításértelmezés*, az USRn függvények feldolgozásához a $030A . . . $030B címeken található *utasításkiértékelés* vektort írjuk át. Erre a következő alkalommal kerül sor.

BARNA LÁSZLÓ

Z80 programok haladóknak Spectrumra és Primóra

8. Manószerkesztő program I.

A manószerkesztő szerkezete

A program törzsének (SPRDEF) inicializáló része (INIT) beállítja néhány rendszerváltozó kezdőértékét, bekéri a manóalakfájl adatait, és megrajzolja a képernyőt (PICT). A program törzsének másik része a billentyűnyomásokat (KEYIN) kiszolgáló rutinokat hívja meg (EXEC). Minden billentyűparancs végrehajtása után újból kiírja a képernyőre a kurzorkoordinátákat.

A program többi része a kiszolgáló rutinok gyűjteménye. Egy ilyen rutin egy billentyűnyomáshoz tartozik, a *táblázat* szerint. A kiszolgáló rutinok nagy részét a következő részben ismertetem.

A program tartalmaz még egy halom alaprutint, ide tartoznak például a 6. részben bemutatott rutinok. Ezeket a kiszolgáló rutinok vagy a program törzséhez tartozó rutinok hívják. Néhány új, eddig még be nem mutatott alaprutin szerepel a *listában*.

A maradék alaprutinok

SPRNO. A manósorszám helyességét vizsgálja meg. Ha kívül esik a megengedett tartományon ((SP1) . . . (SPE)), akkor carry flaggel jelzi a hibát. Ha a sorszám érvényes, HL-ben adja a címét (OLDADD). Ha az illető manó éppen nem látható az „O" ablakban (6. rész) levő automata listában, akkor új automata lista keletkezik (lásd LISTSP).

OLDADD. Ha HL-ben egy manósorszám van, kiszámítja a címét. Az eredmény is HL-be kerül.

ATOB. Ez a fontos rutin az „A" ablakból nagyítva másolja a szerkesztés alatt álló manót a „B" ablakba.

SAVESP. Az SBUFF nevű pufferbe menti a szerkesztés alatt álló manót. Ezt a rutint minden „súlyosabb" következménnyel járó rutin meghívja, mielőtt elrontaná a manót. Ha a parancs kiadása után meggondoljuk magunkat, a D (delete) billentyűvel újra elővarázsolhatjuk az előző állapotot.

PATTNO. Ellenőrzi a mintázat sorszámát. Ha nem esik a megengedett 0 . . . 7 tar-

bill	rutin	hatás
5	LEFT	kurzor balra
8	RIGHT	kurzor jobbra
6	DOWN	kurzor le
7	UP	kurzor fel
0	PLOT	pont invertálás
f	FILLH	mintás feltöltés
o	OLD	manó felhozás a fájlból (OR!)
p	PATT	új mintázat megadás
Enter	NEW	manó beírás a fájlba
b	BACK	old lista vissza mozgatás
n	FORW	old lista előre mozgatás
v	CLEAR	szerkesztőmező törlése
r	ROT	forgatás 90 fokkal
m	TURN	tükrözés függőleges tengelyre
i	INV	invertálás
l	LINE	vonalhúzás a két kurzor közt
h	HELP	angol nyelvű haszn. utasítás
Space	BREAK	visszatérés (basicbe)
s	SIGN	"Y" kurzort az "X"-re állítja
t	RTS	"X" kurzort az "Y"-ra állítja
d	DEL	az utolsó parancs törlése

```
 1 ;-------------------------;
 2 SPRNO - RET  C
 3         LD   A,H
 4         OR   A
 5         SCF
 6         RET  NZ
 7         LD   A,(SPE)
 8         CP   L
 9         RET  C
10         LD   A,(SP1)
11         SUB  L
12         JR   NC,SPRN0
13         CP   -7
14         JR   NC,SPRNE
15         LD   A,L
16         SUB  7
17         JR   SPRN1
18 SPRN0   LD   A,L
19 SPRN1   LD   (SP1),A
20 SPRNE   CALL LISTSP
21         LD   A,L
22         CALL OLDADD
23         RET
24 ;-------------------------;
25 OLDADD  PUSH BC
26         LD   H,A
27         LD   L,0
28         SRA  H
29         RR   L
30         LD   BC,(SPRADD)
31         ADD  HL,BC
32         POP  BC
33         RET
34 ;-------------------------;
35 ATOB    LD   DE,BADD
36         LD   HL,AADD
37         LD   B,SIZ
38 ATOB1   PUSH BC
39         LD   C,SIZ/8
40 ATOB11  LD   A,(HL)
41         CALL ATOB_B
42         INC  HL
43         DEC  C
44         JR   NZ,ATOB11
45         LD   BC,SIZ/8
46         OR   A
47         SBC  HL,BC
48         CALL HLDOWN
49         EX   DE,HL
50         LD   BC,SIZ/2
51         SBC  HL,BC
52         LD   B,4
53 ATOB2   CALL HLDOWN
54         DJNZ ATOB2
55         EX   DE,HL
```

tományba, a carry flaggel kiabál. Egyébként a minta címét adja meg a HL regiszterben.

A programtörzs működése

Ebből egyedül az EXEC rutin említésre méltó. Univerzális, táblázat szerinti szubrutinhívást megvalósító rutin. A TABLE1 táblázat tartalmazza azokat a kódokat, amelyek szerint van hova ugrani. Ezek most a billentyűkódok. A TABLE2 nevű táblázatban a TABLE1 sorrendjének megfelelően ugrási címek vannak.

Érvényes kód esetén elugrik a megfelelő rutinra. Érvénytelen, vagyis a táblázatban nem szereplő kód esetén beáll a zéró flag, és visszatér. Ezt persze csak akkor tesztelhetjük a hívó programban, ha biztosítva van, hogy az EXEC által meghívott összes rutin Zf nélkül tér vissza.

Fontos, hogy a TABLE1 táblázat végjele, egy 0 értékű bájt, ne maradjon le, egyébként érvénytelen kód esetén elszállás várható.

```
56          POP   BC
57          DJNZ  ATOB1
58          RET
59 ATOB_B   LD    B,4
60 ABB1     CALL  ATOB_C
61          INC   DE
62          DJNZ  ABB1
63          RET
64 ATOB_C   PUSH  BC
65          PUSH  DE
66          PUSH  HL
67          LD    B,$02
68 ABC1     RLA
69          RR    C
70          SRA   C
71          SRA   C
72          SRA   C
73          DJNZ  ABC1
74          RRC   C
75          RRC   C
76          RRC   C
77          RRC   C
78          PUSH  AF
79          EX    DE,HL
80          LD    B,4
81 ABC2     LD    (HL),C
82          CALL  HLDOWN
83          DJNZ  ABC2
84          POP   AF
85          POP   HL
86          POP   DE
87          POP   BC
88          RET
89 ;-------------------------------;
90 SAVESP   LD    HL,SBUFF
91 NEWST    LD    DE,AADD
92          EX    DE,HL
93          LD    C,SIZ
94 SASP2    LD    B,4
95 SASP3    LD    A,(HL)
96          LD    (DE),A
97          INC   HL
98          INC   DE
99          DJNZ  SASP3
100         DEC   HL
101         DEC   HL
102         DEC   HL
103         DEC   HL
104         CALL  HLDOWN
105         DEC   C
106         JR    NZ,SASP2
107         RET
108 ;-------------------------------;
109 PATTNO   RET   C
110         LD    A,L
```

```
111         CP    8
112         CCF
113         RET   C
114         ADD   HL,HL
115         ADD   HL,HL
116         ADD   HL,HL
117         LD    BC,PATTS
118         ADD   HL,BC
119         RET
120 ;-------------------------------;
121 SPRDEF  CALL  IN11
122 SPRD1   CALL  KEYIN
123         CALL  EXEC
124         CALL  PRPOS
125         JR    SPRD1
126 ;-------------------------------;
127 PRPOS   CALL  OPEN2
128         CALL  MSG
129         DEFB  AT,10,4,EOL
130         LD    A,(POS)
131         CALL  DEC1A
132         CALL  MSG
133         DEFB  AT,11,4,EOL
134         LD    A,(POS+1)
135         CALL  DEC1A
136         CALL  MSG
137         DEFB  AT,12,4,EOL
138         LD    A,(SPOS)
139         CALL  DEC1A
140         CALL  MSG
141         DEFB  AT,13,4,EOL
142         LD    A,(SPOS+1)
143         CALL  DEC1A
144         RET
145 ;-------------------------------;
146 EXEC     LD    DE,TABLE1
147         LD    HL,TABLE2
148         LD    B,A
149 EXE0     LD    A,(DE)
150         OR    A
151         RET   Z
152         CP    B
153         JR    Z,EXE1
154         INC   HL
155         INC   HL
156         INC   DE
157         JR    EXE0
158 EXE1     LD    A,B
159         CALL  PIP
160         LD    A,(HL)
161         INC   HL
162         LD    H,(HL)
163         LD    L,A
164         JP    (HL)
165 ;-------------------------------;
```

```
166 TABLE1  DEFM  "56/80forEbn"
167         DEFM  "vrmilh std"
168         DEFB  0
169 TABLE2  DEFW  LEFT,DOWN,UP
170         DEFW  RIGHT,PLOT
171         DEFW  FILLN,OLD,PATT
172         DEFW  NEW,BACK,FORW
173         DEFW  CLEAR,ROT
174         DEFW  TURN,INV,LINE
175         DEFW  HELP
176         DEFW  BREAK,SIGN
177         DEFW  RTS,DEL
178 ;-------------------------------;
179 IN11     LD    HL,0
180         LD    (POS),HL
181         LD    (SPOS),HL
182         XOR   A
183         LD    (SP1),A
184         CALL  PICT
185         CALL  SAVESP
186         LD    C,5
187 IN111    CALL  INP
188         DEFM  "Sprite "
189         DEFM  "address? "
190         DEFB  EOL
191         JR    C,IN111
192         LD    (SPRADD),HL
193         LD    C,3
194 IN112    CALL  INP
195         DEFM  "Number of "
196         DEFM  "sprites? "
197         DEFB  EOL
198         JR    C,IN112
199         LD    A,L
200         DEC   A
201         LD    (SPE),A
202         LD    H,L
203         LD    L,0
204         SRA   H
205         RR    L
206         LD    (FILEN),HL
207         CALL  PICT
208         RET
209 ;-------------------------------;
210 PICT     CALL  OPEN2
211         LD    (IY+84),$00
212         LD    (IY+83),$30
213         LD    (IY+87),$00
214         LD    (IY+14),$30
215         LD    A,$06
216         OUT   ($FE),A
217         CALL  CLS
218         CALL  MSG
219         DEFB  AT,0,0,PAP,1
220         DEFB  INK,7
```

A billentyűnyomásokat kiszolgáló rutinok

Nagy részük teljesen kézenfekvő. Nézzünk meg példaként egyet! A LINE rutin az L billentyűhöz tartozik, ki lehet keresni a TABLE1 és TABLE2 táblázatból. Hatása: a két pontkurzort összeköti egy közelítőleg egyenes vonallal, majd az Y kurzort ráállítja az X-re.

```
221          DEFM " SPRITE"
222          DEFM " DEFINER "
223          DEFB CR
224          DEFM "     32*32"
225          DEFB 6,CR,6,CR
226          DEFB PAP,6,INK,0
227          DEFB 6,CR,CR
228          DEFM " Hit H "
229          DEFM "for help "
230          DEFB CR,6,CR
231          DEFM " Address="
232          DEFM "00000  "
233          DEFB CR
234          DEFM "  Lenght="
235          DEFM "00000  "
236          DEFB CR,6,CR
237          DEFM " x =      "
238          DEFB CR
239          DEFM " y =      "
240          DEFB CR
241          DEFM " x'=      "
242          DEFB CR
243          DEFM " y'=      "
244          DEFB CR,6,CR,6,CR
245          DEFB 6,6,6,6
246          DEFB PAP,7,INK,7
247          DEFB EOL
248          LD   BC,#0909
249          LD   D,6
250 PICTB    CALL A1BC
251          CALL MSG
252          DEFM "          "
253          DEFB EOL
254          INC  B
255          DEC  D
256          JR   NZ,PICTB
257          CALL MSG
258          DEFB INK,0,A1,1B,0
259          DEFB EOL
260          LD   C,4
261 SCRL1    LD   B,4
262 SCRL2    CALL MSG
263          DEFB PAP,5
264          DEFM "    "
265          DEFB PAP,7
266          DEFM "    "
267          DEFB EOL
268          DJNZ SCRL2
269          DEC  C
270          JR   NZ,SCRL1
271          CALL CLRIN
272          CALL CLEAR1
273          CALL LISTSP
274          LD   E," "
275          CALL MSG
```

Először meghívja a SAVESP rutint — tehát az utoljára húzott vonal törölhető —, majd felcseréli a két kurzorkoordinátát. (Az RTS hatása ugyanis: (POS):=(SPOS)), majd BC-ben kiszámítja a kezdőpont abszolút koordinátáit és a CDS rendszerváltozóban tárolja le. Ezt használja a múltkor bemutatott DRAW rutin.) Ezután kiszámítja az elmozdulás koordinátáinak abszolút értékét és előjelét. Ehhez nyújt segítséget

```
276          DEFB AT,7,9,EOL
277          LD   HL,(SPRADD)
278          CALL DECIHL
279          CALL MSG
280          DEFB AT,8,9,EOL
281          LD   HL,(FILEN)
282          LD   E," "
283          CALL DECIHL
284          RET
285 ;-------------------------;
286 LINE     CALL SAVESP
287          LD   DE,(POS)
288          CALL RTS
289          LD   (SPOS),DE
290          LD   BC,(POS)
291          CALL ACOORD
292          LD   (CDS),BC
293          LD   A,(POS)
294          LD   B,L
295          CALL ABSSGN
296          PUSH HL
297          LD   A,(POS+1)
298          LD   B,D
299          CALL ABSSGN
300          LD   B,H
301          LD   D,L
302          POP  HL
303          LD   C,H
304          LD   E,L
305          CALL DRAW
306          CALL RTS
307          CALL ATOB
308          RET
309 ABSSGN   SUB  B
310          LD   L,#FF
311          LD   H,A
312          RET  NC
313          NEG
314          LD   H,A
315          LD   L,#01
316          RET
317 ;-------------------------;
```

az ABSSGN nevű rutin. A C és B regiszterbe kerül dx és dy abszolút értéke, az E és D regiszterbe pedig az előjelük (+1 vagy −1).

Ilyen bemenő adatokkal kell hívni az egyenesrajzoló (DRAW) rutint. Utána csak egy RTS (a kurzorkoordináták fel lettek cserélve!) és egy ATOB műveletre van szükség. Mindkettő szerepelt már.

UHERKOVICH PÉTER

● A számítógépes grafika fegyvertárában természetesen ezekre a problémákra is találhatunk megoldásokat. Az egyik lehetőség az ún. spline-ok (szplájnok) módszere, a másik megoldás a francia Bezier nevéhez fűződik. A spline arról a rugalmas vonalzóról kapta a nevét, amelyet például a hajógyári tervezők jól ismernek, és amellyel tetszőleges sima görbék húzhatók úgy, hogy a vonalzót a rajzlapon a kívánt alakra hajlítjuk, majd helyzetét súlyokkal rögzítjük. Az analógia abban rejlik, hogy a spline-oknál is csak egy néhány pontra és esetleg e pontokban a görbe érintőjére van szükségünk, a közbenső pontokat egy alkalmas görbét előállító interpoláló függvényből számítja ki a program.

A Bezier-görbék is hasonló módon származtathatók. Itt az interpoláló függvény egy paraméteresen megadott polinom, amelynek adatai az ún. Bezier-keretet feszítő pontok koordinátáiból számíthatók. Az *1. ábrán* másod-, harmad- és negyedrendű Bezier-keretekre és a hozzájuk tartozó Bezier-görbékre látunk példát. A Bezier-görbék a keret első és utolsó pontjában érintik a keret megfelelő egyeneseit, a keret többi pontjának helyzete pedig valahogyan befolyásolja a görbe alakját.

Anélkül, hogy a matematikai részletekbe belemennénk, a *listán* megadjuk egy harmadrendű Bezier-görbe kiszámítására és felrajzolására szolgáló Pascal-eljárásokat. Ha valakit az elméleti alapok és a kérdéskör egyéb lehetőségei is érdekelnek, figyelmébe ajánljuk a Műszaki Könyvkiadónál 1985-ben megjelent „Interaktív számítógépes grafika" c. könyvet, amelynek szerzői W. M. Newman és R. F. Sproull. A hivatkozott eljárásokat kis módosítással mi is innen vettük át.

A harmadrendű Bezier-görbéhez három egyenesből álló keret tartozik, négy ponttal. A pontok koordinátáit a D tömbben tároljuk, melynek első indexe a pont sorszáma, második indexe pedig a koordináta: 1 az x, 2 az y koordináta.

A BEZIER-eljárás a görbe U paraméteréből (melynek értéke az első pontban 0, az utolsóban pedig 1) kiszámítja a B súlyfüggvény segítségével a hozzá tartozó X és Y koordinátát. A súlyfüggvényeket a BSULY eljárás számítja: minden ponthoz egy-egy súlyfüggvény tartozik. A CC függvényt a

Számítógépes grafika Pascalban

BEZIER-GÖRBÉK

Az előző cikkekből már ismerjük, hogyan lehet egyenest, kört, ellipszist rajzolni. Ez mind szép, de hát ezeken kívül még annyiféle görbe létezik! Felrajzolásukhoz mind külön algoritmust kell találnunk? És mit tehetünk akkor, ha nem szabályos görbékkel akarunk valamit ábrázolni, hanem olyasmit szeretnénk végrehajtani a számítógéppel, amit egyébként a hagyományos grafikában szabadkézi rajzzal oldunk meg?

BSULY eljárás használja állandóinak meghatározására.

A GORBERAJZOLAS eljárás az így meghatározott Bezier-függvény képét felrajzolja úgy, hogy a görbét 40 szakaszra bontja, és ezeket a szakaszokat most már egyenesekkel helyettesiti.

Talán jobb, ha nem mondjuk meg, mi a Bezier-görbék előnye, hanem az olvasó maga győződik meg róla: bebillentyűzi Spectrumába a mellékelt mintaprogramot. Ez persze ismét felhasználja a korábban már ismertetett, egyenesrajzoló eljárásokat tartalmazó LINE nevű include-fájlt.

A mintaprogram segítségével szabadon rajzolhatunk a képernyőre. Ilyen rajzolóprogram a Spectrumra sok és kiváló készült, de ehhez hasonló alapötletre egyik sem épül. A program betöltése után a képernyő jobb oldalán egy menü jelenik meg (ez természetesen a T opcióval forditott változatra érvényes). Ha megnyomjuk például az "5" billentyűt, az eddig üres részen egy Bezier-keret lesz látható. A pontokat a képernyőn belül az "5", "6", "7", "8" kurzorvezérlő gombokkal lehet mozgatni, hogy éppen melyiket, azt az dönti el, melyik gombot nyomtuk meg az "1", "2", "3" és "4" közül. Hogy melyik sorszám melyik ponthoz van rendelve, arról pedig legegyszerűbben próbálkozással győződhetünk meg.

Ha a "G" gombot nyomjuk meg, a Spectrum felrajzolja a képernyőre a kerethez tartozó Bezier-görbét. Ha továbbmozdítjuk valamelyik pontot, a görbe eltűnik, csak a keret jelenik meg újból. Így kisérleti görbéket rajzolhatunk, és ezt büntetlenül addig ismételhetjük, amig az eredménnyel elégedettek nem leszünk. Ekkor nyomjuk meg az "O" gombot (O. K.!), mire a görbe képe ismét megjelenik, de már véglegesen: ha a keretet elmozdítjuk, a görbe akkor is a képernyőn marad.

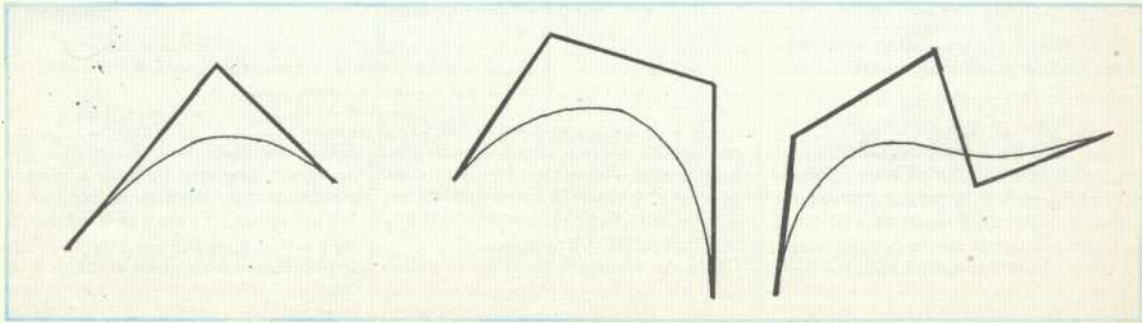
A menü K opciójával a keret rajzát is véglegesithetjük. Kis gyakorlással — és természetesen kézügyességgel — a program segitségével úgy rajzolhatunk a képernyőre, mint egy darab papirra ceruzával. Erre jó példa a *2. ábrán* látható repülőgép rajza, amelyet a cikk szerzője ügyetlenkedett ki. Ha nem tetszik, amit rajzoltunk, a "R"adír opcióval az egész képet kitörölhetjük.

A képernyőről hardcopy is készíthető, és a menüt a kép háttértárra való kimentése, illetve onnét beolvasása egészíti ki. A háttértár lehet kazettás magnó vagy lemezmeghajtó; ha ez utóbbit szeretnénk elérni, a fájlnév, amelyre a program rákérdez, a meghajtó számával kezdődik, majd kettőspont következik, és ez után tetszőleges négybetűs sztring áll. Vigyázzunk, mert az utolsó karakter beütése után a kivánt háttértárművelet automatikusan beindul!

Nézzük meg most a program érdekesebb részleteit! A próbarajzolást a rejtett, második képernyőfájl használatával oldjuk meg. Ez azt jelenti, hogy definiálunk egy képernyőnyi memóriának megfelelő karaktertömböt (KEP típus), és ebbe csak akkor másoljuk át a látható képernyőfájl tartalmát, ha az O vagy a K opciót használjuk, vagy ha háttértárból kellett valamit beolvasni. A keret minden egyes módositása előtt a rejtett képet először bemásoljuk a látható kép helyére, csak ezután rajzoljuk rá a keret vagy a görbe ideiglenes képét. Ez az interaktiv grafika egyik egyszerű, de hatásos módszere.

A CLWINDOW eljárás olyan alfanumerikus ablakot töröl, amelynek bal felső sarka az X-edik oszlopban és Y-adik sorban helyezkedik el, Z sor mélységű és W karakter szélességű. Ez az eljárás abban különbözik a PAGE-től, hogy az attributumokat

1. ábra



```
 10 PROGRAM BezierGorbe;
 20 {$L-}
 30 {Keszitette: dr Koboldy Peter 1986.}
 40 CONST N=4;M=215;
 50 TYPE KEEP=ARRAY[0..28,0..31,0..7]OF CHAR;
 60 VAR I,J,L:INTEGER;
 70     D:ARRAY[0..3,1..2]OF REAL;
 80     K:KEEP;
 90     NEV:ARRAY[1..8]OF CHAR;
100
110 PROCEDURE SPOUT(C:CHAR);
120 BEGIN
130  INLINE($FD,$21,$3A,$5C,$DD,$7E,2,$D7)
140 END; {SPOUT}
150
160 PROCEDURE AT(X,Y:INTEGER);
170 BEGIN
180  SPOUT(CHR(22));
190  SPOUT(CHR(X));SPOUT(CHR(Y))
200 END; {AT}
210
220 PROCEDURE CLWINDOW(X,Y,Z,W:INTEGER);
230 VAR I,J:0..31;
240 BEGIN
250  FOR I:=0 TO Z DO
260  BEGIN
270   AT(X+I,Y);
280   FOR J:=1 TO W DO
290   WRITE(' ')
300  END
310 END; {CLWINDOW}
320
330 {$L+}
340 {$F 1:LINE  }
350
360 FUNCTION CC(N,I:INTEGER):INTEGER;
370 VAR J,A:INTEGER;
380 BEGIN
390  A:=1;
400  FOR J:=I+1 TO N DO A:=A*J;
410  FOR J:=1 TO N-I DO A:=A DIV J;
420  CC:=A
430 END; {CC}
440
450 FUNCTION BSULY(I,N:INTEGER;U:REAL):REAL;
460 VAR J:INTEGER;V:REAL;
470 BEGIN
480  J:=CC(N,I);V:=J;
490  FOR J:=1 TO I DO V:=V*U;
500  FOR J:=1 TO N-I DO V:=V*(1-U);
510  BSULY:=V
520 END; {BSULY}
530
540 PROCEDURE BEZIER(VAR X,Y:REAL;U:REAL;N:INTEGER);
550 VAR I:INTEGER;B:REAL;
560 BEGIN
570  X:=0;Y:=0;
580  FOR I:=0 TO N DO
590  BEGIN
600   B:=BSULY (I,N,U);
610   X:=X+D[I,1]*B;Y:=Y+D[I,2]*B;
```

```
620  END
630 END; {BEZIER}
640
650 PROCEDURE GORBERAJZOLAS;
660 VAR I:INTEGER;X,Y,X1,Y1:REAL;
670 BEGIN
680  FOR I:=0 TO 40 DO
690  BEGIN
700   BEZIER(X,Y,I/40,3);
710   IF I<>0 THEN
720   PixelLine(ROUND(X),ROUND(Y),
730    ROUND(X1),ROUND(Y1));
740   X1:=X;Y1:=Y
750  END
760 END; {GORBERAJZOLAS}
770
780 PROCEDURE RULER;
790 VAR I:INTEGER;
800 BEGIN
810  J:=J+1;IF J=5 THEN L:=8;
820  POKE($4000,K);
830  FOR I:=1 TO 3 DO
840   PixelLine(ROUND(D[I,1]),
850   ROUND(D[I,2]),
860   ROUND(D[I-1,1]),
870   ROUND(D[I-1,2]))
880 END; {RULER}
890
900 PROCEDURE HEADER;
910 VAR I,J:INTEGER;A:CHAR;
920 BEGIN
930  FOR I:=1 TO 8 DO NEV[I]:='@';
940  FOR J:=1 TO 2000 DO;
950  AT(19,27);WRITE('NEV:');AT(21,27);
960  A:=CHR(0);I:=1;
970  WHILE I<6 DO
980  BEGIN
990   WHILE(ORD(A)<12) OR ((ORD(A)<48)
1000  AND (ORD(A)>12) OR (ORD(A)>90)) DO
1010  A:=INCH;
1020  IF A=CHR(12) THEN
1030  BEGIN
1040   I:=I-2;AT(21,27+I);WRITE(' ');AT(21,27+I)
1050  END;
1060  IF A<>CHR(12) THEN
1070  BEGIN
1080   WRITE(A);NEV[I]:=A
1090  END;
1100  A:='@';
1110  I:=I+1;
1120  FOR J:=1 TO 3000 DO
1130 END
1140 END; {HEADER}
1150
1160 PROCEDURE MENU;
1170 TYPE STRING=ARRAY[1..4]OF CHAR;
1180
1190 PROCEDURE INVERSE(I:INTEGER);
1200 VAR T,P:INTEGER;
1210 BEGIN
1220  IF I=0 THEN BEGIN T:=0;P:=7 END
```

is törli. A RULER eljárás a keretet rajzolja fel. A HEADER eljárás rákérdez a fájlnévre és beolvassa. A "S"ave és a "L"oad utasítás aktivizálásakor kerül sor a hívására.

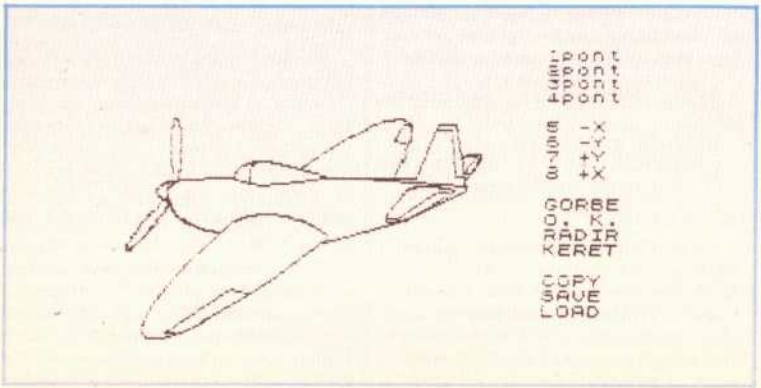
A MENU eljárás, mint a neve is mutatja, a menü kiírására szolgál. A JOYSTICK eljárás a menüopciók kiválasztását irányitja. Ez az eljárás az egész program magja, ez gondoskodik a többi eljárás hívásáról is. Végül a néhány sornyi főprogram kezdőértéket ad a keretpontok koordinátáinak, és végtelen ciklusban hívja a JOYSTICK és a GORBERAJZOLAS rutinokat.

Még egy apróságra hívjuk fel olvasóink figyelmét. A program akkor dolgozik hatékonyan, ha a keret mozgatásának legkisebb egysége két képpont távolsága. Ez a pontosság azonban igen lelassítja a keret elmozditását. Az ellentétes követelményeknek úgy tud eleget tenni a program (és még csak nem is bonyolitjuk a kezelést!), ha a következő működést valósítja meg. Ha valamelyik kurzormozgató gombot folyamatosan lenyomva tartjuk, a program elkezdi

```
1230   ELSE BEGIN T:=7;P:=0 END;
1240   SPOUT(CHR(16));SPOUT(CHR(T));
1250   SPOUT(CHR(17));SPOUT(CHR(P));
1260 END; {INVERSE}
1270
1280 PROCEDURE P(I:INTEGER;K:CHAR;S:STRING);
1290 BEGIN
1300   AT(I,27);INVERSE(1);WRITE(K);
1310   INVERSE(0);WRITE(S)
1320 END; {P}
1330
1340 BEGIN
1350   CLWINDOW(0,27,21,5);
1360   P(0,'1','pont');
1370   P(1,'2','pont');
1380   P(2,'3','pont');
1390   P(3,'4','pont');
1400   P(5,'5',' -X ');
1410   P(6,'6',' -Y ');
1420   P(7,'7',' +Y ');
1430   P(8,'8',' +X ');
1440   P(10,'G','ORBE');
1450   P(11,'0','. K.');
1460   P(12,'R','ADIR');
1470   P(13,'K','ERET');
1480   P(15,'C','OPY ');
1490   P(16,'S','AVE ');
1500   P(17,'L','OAD ');
1510   K:=PEEK(#4000,KEEP)
1520 END; {MENU}
1530
1540 PROCEDURE JOYSTICK;
1550 VAR A:CHAR;
1560 BEGIN
1570   A:='0';
1580   J:=0;L:=1;
1590   WHILE A<>'0' DO
1600   BEGIN
1610     A:=INCH;
1620     CASE A OF
1630       CHR(0):BEGIN
1640                J:=0;L:=1
1650              END;
1660       '1':I:=0;
1670       '2':I:=1;
1680       '3':I:=2;
1690       '4':I:=3;
1700       '5':IF D[I,1]>=L THEN
1710           BEGIN
1720             D[I,1]:=D[I,1]-L;RULER
1730           END;
```

```
1740       '6':IF D[I,2]>=L THEN
1750           BEGIN
1760             D[I,2]:=D[I,2]-L;RULER
1770           END;
1780       '7':IF D[I,2]<=191-L THEN
1790           BEGIN
1800             D[I,2]:=D[I,2]+L;RULER
1810           END;
1820       '8':IF D[I,1]<=M-L THEN
1830           BEGIN
1840             D[I,1]:=D[I,1]+L;RULER
1850           END;
1860       'R':BEGIN
1870             PAGE;MENU;K:=PEEK(#4000,KEEP)
1880           END;
1890       'K':BEGIN
1900             POKE(#4000,K);RULER;K:=PEEK(#4000,KEEP)
1910           END;
1920       'G':GORBERAJZOLAS;
1930       'C':BEGIN
1940             CLWINDOW(0,27,21,5);USER(#0EAC);MENU
1950           END;
1960       'S':BEGIN
1970             HEADER;
1980             TOUT(NEV,ADDR(K),SIZE(K));MENU
1990           END;
2000       'L':BEGIN
2010             HEADER;TIN(NEV,ADDR(K));POKE(#4000,K);MENU
2020           END
2030       END
2040   END
2050 END; {JOYSTICK}
2060
2070 BEGIN
2080   PAGE;
2090   MENU;
2100   D[0,1]:=0;   D[0,2]:=0;
2110   D[1,1]:=20;  D[1,2]:=191;
2120   D[2,1]:=M-20;D[2,2]:=191;
2130   D[3,1]:=M;   D[3,2]:=0;
2140   I:=0;
2150   WHILE 1<>2 DO
2160   BEGIN
2170     JOYSTICK;
2180     POKE(#4000,K);
2190     GORBERAJZOLAS;
2200     K:=PEEK(#4000,KEEP)
2210   END
2220 END.
>C
```

2. ábra



számlálni az egymást követő ismétléseket, és a negyedik után a lépésközt automatikusan nyolc képponttávolságra váltja át. Így nagy távolságokat is gyorsan be tudunk járni. Ha pontos megközelítésre van szükség, a kurzormozgató gombot időnként el kell engedni. Ekkor a lépésköz visszavált egyre.

Ennek a hosszadalmasan leírt műveletnek az algoritmusa igen egyszerű: a RULER eljárás első sorára terjed csak ki. I az ismétlések száma, L pedig a lépésköz.

Ismét hagyunk az olvasónak önálló feladatot: megoldhatja például vastagabb vonalak vagy mintás vonal (szaggatott/pontvonal stb.) hívását a menüből.

DR. KABOLDY PÉTER

Függvények és utasítások III.

Függvények vagy eljárások

A cikkünk első részében leírtak után joggal merülhet fel az olvasóban a kérdés: ha a „tiszta" függvények olyan szépek, miért alkalmaznak mégis a programozási nyelvek mindenféle más „piszkos" eszközt? A válasz kettős: a hatékonyság az egyik, az emberi tényező a másik ok.

Nézzük először végig, melyek is azok a bizonyos eszközök, amelyeket a programozási nyelvek olyan előszeretettel alkalmaznak. Az első és legfontosabb: az ún. *függvény mellékhatással*, más néven az *értékvisszaadó eljárás*. Azokat a valamiket (függvényeket vagy eljárásokat) hívják így, amelyek egyrészt értéket adnak vissza, másrészt a program állapotát is módosítják.

Ez is, az is...

Vannak olyan nyelvek — nem is kevesen —, amelyek fölöslegesnek tartják két különböző konstrukciós eszköz, a függvény és az eljárás bevezetését, hiszen olyan nagy a hasonlóság közöttük. Oldjuk meg egyetlen eszközzel, ami mindkettő tulajdonságait egyesíti. Tény, hogy ez egy univerzális valami: nappal csapszék — éjjel csapágy. Lehet használni tiszta függvények helyett és tiszta eljáráshívás helyett (a kapott értéket nem kell felhasználni), és úgy is, ahogy az előző kettőt nem lehetett.

Tipikusan ilyen például egy függvény, amelyet nevezzünk el NEXT-nek: egy adott listából kiveszi a következő elemet (ezt adja vissza), és közben a mutatót, hogy hányadik elemnél tartunk, eggyel növeli. Látható, hogy ez az eljárás még utasítást is megtakaríthat ahhoz képest, ha az eljárás egy változó paraméterében adná vissza a következő elemet, hiszen azt onnan elő kell venni, ha valamit csinálni akarunk vele.

Tudjuk azonban, hogy nincsen rózsa tövis nélkül: a mellékhatással járó függvénynek is megvannak a maga hátrányai. Ezeket a hívásokat nyilván ott használjuk, ahol eddig csak függvényeket alkalmaztunk, például egy másik függvény argumentumaként. Eddig erről mindössze annyit mondtunk, hogy a függvény kiszámítása előtt kiszámítjuk az argumentumok értékét, de semmit nem mondtunk arról, hogy ezt milyen sorrendben tesszük, és ha két argumentumhoz azonos kifejezés tartozik, azt kétszer számoljuk-e ki vagy csak egyszer. Ez egyébként a fordítóprogramok optimalizálási stratégiájának egyik kedvenc célpontja. Könnyű kitalálni olyan F(x,y) függvényt, amelynek nem mindegy, hogy az F(NEXT,NEXT) hívás esetén az első argumentumban egy lista 3. elemét kapja és a másodikban a 4. elemét vagy fordítva, netán mindkettőben a 3. elemet.

A sorrendiség

A programozási nyelvek ezzel a problémával szemben kétféle álláspontot képviselnek. Egy részük szigorúan előírja a függvényhívásokban az argumentumok kiszámítási sorrendjét és ezáltal az aritmetikai kifejezések kiszámítását is, más részük (például az Ada) azt mondja: galád cenk az a programozó, aki olyan programot ír, amelynek az értéke ettől függ — és így elhárítja magáról a felelősséget.

Annak idején az ALGOL60 nyelv szinte rájátszott az első megoldásra azzal, hogy bevezette a *név szerinti paraméterhívást* (call by name), ahol a paraméter értékét mindig újra ki kellett számítani, valahányszor a paramétert felhasználták. Látható, hogy ez egyszerű függvények esetén azonos a szükség szerinti paraméterkiszámítással, hiszen a függvény a külvilágot nem változtathatja meg, és ezért az argumentumban lévő függvényhívás mindig azonos eredményt ad. Mellékhatásos függvények esetén viszont elég furcsa dolgok történhetnek. Ha például egy ciklussal végigfutunk egy tömb indexén, akkor egyetlen paraméterrel elérhetjük a tömb minden elemét.

```
REAL PROCEDURE Skalárszorzat (a,b,n);
  INTEGER n; REAL a,b;
  BEGIN FOR i:=1 STEP 1 UNTIL n DO
  Skalárszorzat:=Skalárszorzat+a*b;
  END Skalárszorzat;
```

Ha az eljárást Skalárszorzat(A [i], B [i], k) alakban hívjuk meg, ahol k a két vektor közös hossza, akkor a FOR ciklus hatására az i változó 1-től k-ig változik, az a és b változó pedig felveszi az A és a B tömb aktuális i-edik elemének értékét, és ezek összegződnek.

Eljáráshívás paraméterátadással

Egy másik ilyen „zavaros ügy" szintén a paraméterátadással kapcsolatos, de az eljáráshívásoknál általában jelentkezik. Természetes dolog, hogy egy eljárás nem egyetlen változót módosít, hanem többet. Tegyük fel, hogy van egy eljárásunk, amely két paraméterét módosítja: az egyikhez hozzáad kettőt, a másikhoz hármat. Például:

```
Növel(VAR:x,y:INTEGER)  BEGIN
x=x+2; y=y+3 END
```

Mi történik akkor, ha az eljárást úgy hívjuk meg, hogy mindkét paraméter helyére ugyanazt a változót írjuk be? Például Növel(I,I). Kettővel lesz nagyobb a közös aktuális paraméter vagy hárommal, netán öttel?!

Az eljárások paraméterátadására két stratégia van forgalomban. *Érték szerinti paraméterátadásnak* nevezzük azt a módot, amikor az eljárás meghívása előtt a paraméter értékét kivesszük a változóból, elrakjuk egy paraméter számára fenntartott helyre, az eljárás alatt ezzel számolunk, ezt módosítjuk, majd a végén visszaírjuk a változóba.

Cím szerintinek nevezzük a paraméterátadásnak azt a módját, amikor a paraméternek csak a címét állapítjuk meg a hívás előtt, és ezen a címen keresztül magát a hívásban szereplő változót kezeljük, valahányszor csak az eljárás hozzányúl.

Világos, hogy az eredmény attól függ, milyen volt a paraméterátadás mechanizmusa, illetve az érték szerinti hívásnál milyen sorrendben adjuk vissza a paramétereket.

Általában véve a gondot az okozza, hogy ugyanahhoz a változóhoz több úton is hozzáférhetünk, és hiába állapítottuk meg róla az egyik percben, hogy értéke valamennyi, mert mire a következő alkalommal hozzáférünk ugyanazon az úton, lehet, hogy értéke teljesen más lesz. Ez bizony elég kellemetlen tulajdonság. Az irodalom *álnevezésnek* (aliasing) hívja.

Az álnevezés elkerülésére a metodológia azt ajánlja, hogy soha ne használjunk egy eljárásban külső változót, amely paraméter is lehet, és két kimenő paramétert soha ne hívjunk meg azonos változóval. No de könnyű ilyet mondani, annál nehezebb el-

lenőrizni. Nem is igen tudok olyan fordítóprogramról, amelyik megtenné.

A „tiszta" függvények „szeplői"

Mint a fenti példákból is látható, az eljárások és a mellékhatással járó függvények használata felvet olyan problémákat, amelyek a „tiszta" függvények esetén nem lépnek fel. Nézzük viszont a másik oldalt.

Az első és legfontosabb probléma talán a *hatékonyság*. A tapasztalat azt mutatja, hogy ugyanaz a program pusztán függvényekkel megírva jóval lassabb lesz és sokkal több memóriát használ fel, mint ha utasításokkal írnánk meg.

Ennek alapvető oka, hogy a függvényhívás a paraméterátadással együtt és a függvényből való visszatérés — különösen, ha összetett adatstruktúrát kell visszaadni értékként — meglehetősen sok időt és állandó memória-karbantartást igénylő tevékenység. A függvényszerű nyelvekben nincsenek felülírható változók, csak paraméterek vannak, amelyeket nem lehet megváltoztatni. Ezért ha egy paraméter értékét módosítva akarjuk felhasználni, új példányt kell belőle készíteni, miközben a régi megmarad. Különösen fogyasztja a memóriát ez az eljárás tömbök, listák, rekordok esetén, amikor néha egyetlen komponens megváltoztatásáért az egész adatstruktúrából új példányt kell létrehozni.

A probléma megszűnését egyesek új elvű, új felépítésű gépektől várják. Az utóbbi időben tényleg meg-is jelentek a piacon olyan ún. LISP gépek, amelyek alkalmasabbak a LISP, a PROLOG és a függvényszerű nyelvek végrehajtására, mint a korábbi gépek voltak.

A másik szempont *az emberi tényező*. Pszichológiai vizsgálatok azt mutatták, hogy az emberek képesek igen hosszú elágazás nélküli utasítássorozatokat egyből megérteni, de a több mélységben egymásba ágyazott, zárójeles kifejezések megértése általában nehézségeket okoz. Márpedig a függvényszerű nyelvekben a zárójelek időnként annyira elszaporodnak, hogy egyesek egy külön jelet is bevezetnek, amelynek jelentése: most zárj be minden megnyitott zárójelet. Persze némi tréninggel javítható a zárójeles kifejezések olvasásának képessége.

A függvényszerű nyelvet sokszor úgy teszik a gyakorlat számára használhatóvá, hogy utasításszerű elemeket vezetnek be, vigyázva arra, hogy a függvény lényegét megőrizzék, vagyis hogy a függvény csak paraméterein keresztül kapjon információt a külvilágról, és csak a visszaadott eredménnyel befolyásolja azt.

FARKAS ERNŐ

Bináris fák

Alapfogalmak

Az utóbbi években viszonylag kevés szó esett az adatfeldolgozásban nagy fontosságú fastruktúrákról. Három részből álló leírásomban az adatbázis-kezelők lelkét, a *kiegyensúlyozott bináris fát* mutatom be a BASIC interpreterek alkalmazóinak. Az igényességet elsősorban az egyszerű megközelítésben tartom szem előtt. Az elméleti leírások után egy Commodore 64-re írt BASIC nyelvű megvalósítást mutatok be. Ezt több mint két éve alkalmazom programjaim eljárásaként a C64 és az MO8X gépen.

Adatszerkezetek

Az adatfeldolgozás klasszikus példája szerint valamilyen azonosító és az ahhoz kapcsolódó információ (bejegyzés) tárolását, keresését, átírását, esetleg megszüntetését kell elvégeznünk egy képzeletbeli táblázatban. A számítógép tárolójának tartalma bájtok (vagy szavak) sorozata. Végső soron a táblázat „bejegyzéseit" is hasonló sorozatba kell szervezni, tudva azt, hogy természetesen az azonosító részhez lényegesen többször kell hozzáférnünk, mint az információs részhez. Mivel a számítógépek RAM-ja a háttértárolókhoz képest kicsi, úgy célszerű megszervezni az adatok tárolását, hogy a feldolgozás során csak az azonosítók, továbbá a megfelelő információs részre mutató adatok kerüljenek a memóriába, az információs rész pedig a háttértár egy direkt elérésű állományát képezze. Az adatfeldolgozás hatékonysága ezek után elsősorban az azonosító táblázaton elvégezhető műveletek hatékonyságától függ.

Az elméleti példákban e megosztástól eltekintünk; feltételezzük, hogy a bejegyzés csak azonosító részből áll. A továbbiakban az azonosító, elem, bejegyzés fogalmakat szinonim fogalmakként használjuk.

Listák

Legegyszerűbb eset, amikor a bejegyzéseket előfordulásuk sorrendjében írjuk fel, és ezután kereséskor szép sorjában végignézzük a táblázat „sorait", elemeit, míg csak rá nem bukkanunk a keresettre. Az így létrehozott adathalmazt *(lineáris) listának* nevezzük. Sajnos n elemű lista esetén átlagosan $n/2$ összehasonlítást kell végezni egy táblázati elem megtalálásához, a létezés kizárásához pedig n összehasonlítás szükséges. Más a helyzet, ha a lista rendezett. Ilyenkor az egyik leghatékonyabb keresési módszer a *bináris keresés:* itt az eléréshez szükséges összehasonlítások száma $\log_2 n$ nagyságrendű. Ezt azonban meg kell hogy előzze egy rendezés, amely jó esetben n^2 nagyságrendű. A rendezést mindenkor el kell végezni, amikor a lista módosul, de lehetséges a beépítésekkel egyidejűleg is.

Láncok

A *lánc* annyiban különbözik a listától, hogy az azonosító mellett van egy ún. *pointer*, amely a következő azonosítóra *(láncszemre)* mutat. Az azonosítók eredeti sorrendje megmarad. Az *1. ábra* egy 6 elemből álló azonosítósorozat tömbbel való ábrázolását mutatja lista és lánc alakban. A lánc elejét a 0. sorban lévő index jelzi. A lánc szemeinek sorrendje: 2, 5, 3, 4, 6, 1. Végig bejárva a láncot (a lánc végét jelzi az *eol* szimbólum) az érintett azonosítók sorszámait betöltjük egy tömbbe, majd e tömb elemeivel indexelt azonosítókon végezzük el a bináris keresést. A lánc felépítése igen nehézkes, és a helyzet ugyan javult valamit a listához képest, de a műveletek számában semmit sem nyerünk.

Több módszer irányul a láncstruktúra hatékonyságának fokozására, például a hash-kódolás.

	lista		lánc	
sorszám	azonosító	sorszám	azonosító	következő
0		0	(lánc eleje)	2
1	körte	1	körte	eol
2	alma	2	alma	5
3	barack	3	barack	4
4	datolya	4	datolya	6
5	ananász	5	ananász	3
6	dió	6	dió	1

1. ábra

A bináris fa fogalma és ábrázolása

A fa a láncnak egy szerkezeti kiterjesztése. A láncban minden azonosítónak legfeljebb egy rákövetkezője lehet. A *bináris fa* csak annyiban különbözik a lánctól, hogy itt egy helyett két rákövetkező is lehet. Ezeket az azonosító bal oldali követőjének (baljának), illetve jobb oldali követőjének (jobbjának) nevezzük. A kezdőelem a fa gyökere.

A láncot és a bináris fa szerkezetét, gráffal való ábrázolását a *2. ábra* mutatja az előző példát felhasználva.

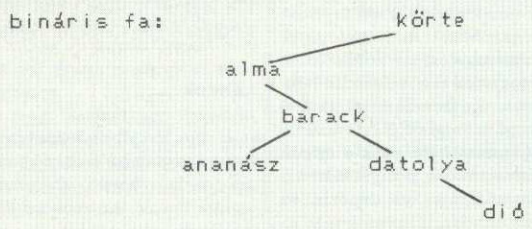
A láncnak egyetlen vége van, az adatok kereséséhez a láncban egyetlen valódi út vezet. Ez az út olyan hosszú, ahány elem van a láncban. A bináris fában azonban általában sok út kínálkozik; egy elem megkereséséhez ezek közül egyetlen jól meghatározott utat kell bejárni. Ez szerencsés esetben csak néhány elemet tartalmaz az összes közül. A továbbiakban a fa elnevezésén a bináris fát értjük. A fát is ábrázolhatjuk tömbbel. Ezt mutatja a *3. ábra.*

Meg kell jegyezni, hogy a lánc és a fa fenti leírása nem tekinthető definíciónak. Ha nem teszünk olyan megszorításokat, amelyek a végtelen ciklust kizárják, akkor nincsenek kizárva például a gyűrűs szerkezetek.

A bináris fa pontos megadásához szükség van a fastruktúrák egy fontos tulajdonságának, a rekurziónak az ismeretére.

Rekurzió

Akkor mondjuk valamiről, hogy *rekurzív, ha önmagát valamilyen módon tartalmazza.* Például bizonyos számsorozatok, körsor vagy akár a szőlőfürt is rendelkezik a rekurzivitás tulajdonságával. A matematikában és a számítástechnikában is felismerték a rekurzió előnyét, és több fontos vonatkozásban is alkalmazzák a *rekurzív definíciót:* többek között adatszerkezetek leírására. Rekurzív definíció például a természetes számok esetében:



2. ábra

3. ábra

sorszám	azonosító	balja	jobbja	
0		1		gyökér indexe
1	körte	2	0	csak balja van
2	alma	0	3	
3	barack	5	4	
4	datolya	0	6	
5	ananász	0	0	levél
6	dió	0	0	

– egy természetes szám
= természetes szám rákövetkezője természetes szám.

A rekurzív tulajdonságokkal rendelkező halmazokra vonatkozó állítások bizonyításának gyakran használt eszköze a teljes indukció.

Ugyancsak népszerűek a *rekurzív algoritmusok* is. Ezek olyan eljárások, amelyek végrehajtásuk során vagy közvetlenül önmagukat, vagy olyan más eljárást hívnak, amely tartalmaz a hívó eljárásra vonatkozó hivatkozást.

A rekurzív definíció és algoritmus előnyéről és hátrányáról igen megoszlóak a vélemények. Kétségtelenül előnyös, hogy például a rekurzív módon megfogalmazható adatstruktúra világosabban és tömörebben kezelhető rekurzív algoritmus segítségével, mint anélkül.

A *rekurzív hívásokkal* igen óvatosan kell bánni! Gondoskodni kell arról, hogy a rekurzív algoritmus befejeződhessen, vagyis egy maximális mélységet ne léphessenek túl a rekurzív hívások. Rekurzív híváskor meg kell szervezni minden mélységhez a

— visszatérési cím megőrzését — ezt jóformán minden programozási nyelv támogatja,

— lokális változók tartalmának és egyéb fontos állapotjelzőknek a megőrzését az adott szintre való visszatérésig.

A fenti két követelményt együtt nem teljesíti például a BASIC nyelv. A Pascal és a FORTH megengedi a rekurzív hívást (az utóbbi kicsit nehézkesebben). A LOGO nyelv viszont maximálisan támogatja a rekurzív hívásokat.

Tömbök, ciklus- és ugróutasítások használatával a rekurzió — általában nagy nehézségek árán — BASIC-ben is megszervezhető.

Bináris fák megadása rekurzív definícióval

Mivel a számítógép tárolói végesek, csak véges alaphalmazú fastruktúrák jöhetnek szóba. Legyen T egy véges halmaz. T bináris fa, ha

(1) [ ], azaz üreshalmaz, vagy

(2) létezik egy olyan t eleme és $T1,T2$ részhalmaza T-nek, hogy

$-T-[t]=T1+T2$

$=T1*T2=[\ ]$

$\equiv T1$ és $T2$ bináris fa

A T halmaz elemeit a fa *csúcspontjainak*, a kiválasztott t csúcspontot a T fa *gyökerének* nevezzük. (Folytatjuk.)

FLENDER ANDRÁS

ADOK-VESZEK-CSERÉLEK

Ebben a rovatban rövid, szöveges, a mikroszámítógépekkel kapcsolatos hirdetéseket közlünk. A díjszabás: közületeknek gépelt soronként (60 karakter) 100,- Ft, magánszemélyeknek az első sor 50,- Ft, minden további sor 20,-Ft. Az NJSZT tagjainak az első három sor ingyenes. Hirdetéseiket a szerkesztőség címére várjuk.

ADOK

C16, Plus/4 programok eladók, 10 Ft/db utánvéttel. Tel.: 06 /1/ 429-168

C16, Plus/4 gépekre három lépésig használható mattfejtőt, kazettán utánvéttel elküldöm címére. Ára: 300 Ft. Keresek C Plus/4-re 12 kb-osnál nagyobb sakkprogramot. Kapta Károly, Csenger, Jókai köz 2. 4765

C64-es játékprogramok nagy számban, olcsón - kívánságra leírással - kaphatók. Tel.: 891-200

Spectrum (új, 48 k) programokkal, sok irodalommal, tartozékokkal eladó. Kiss János, Tatabánya, Kossuth-kert 78. 2800

ZX81 (64 k) programokkal és szakkönyvekkel eladó. Árajánlatot levélben kérek. Szabó Gábor, Siófok, Hársfa u. 13. 8609

ZX81 (16 k) bővítővel együtt, vagy külön-külön eladó. ifj. Hámori György, Kazincbarcika, Pollack M. u. 33. III/1. 3700

CSERÉLEK

Commodore 64-es programokat cserélek, elsősorban kazettán. Mindenféle program érdekel. Válaszokat lehetőleg listával kérek. Címem: Magyar Attila, Kapuvár, Lenin u. 10. 9330

C64-es játékprogramokat cserélek, kazettán. Balázs Roland, Komárom, Tópart u. 6/B. fsz. 2. 2900

Commodore 64 programokat, dokumentációkat cserélek. Keresem Vezetői grafika dokumentációját. Takács Tamás, Csókakő, Dózsa György u. 27. 8074

C16, Plus/4 számítógépre programokat cserélek, Lérant Attila, Zalaapáti, Ady Endre u. 6. 8741

Enterprise programokat cserélek. Listát kérek! Veresegyház, Pf.:28, 2112

VC20 tulajdonosokkal leveleznék programcsere céljából. 32 k-bővítős programok is érdekelnek. Különösképp keresem szintén VC20-ra az "IMPOSSIBLE MISSION" és a "GRÓF" című programokat (32 k). Levélcím: Szabó Csaba, Gyula, Petik A. u. 41. 5700

A HCC Commodore Klub előadásai az 1988. tavaszi félévben

Február 11., 25.: Beszámoló az amerikai klubújdonságokról

Március 10.: (nemcsak Commodore-klubok alapján) Comal-kazetta hardver

Március 24.: Comal-kazetta szoftver

Április 14.: Egy különleges bővítőkártya C64/C128 számára

Április 28.: A bővítőt kezelő szoftver

Május 12., 26.: Adatkezelők

Az előadások mellett konzultáció, információcsere, szaklapok olvasási lehetősége, nemzetközi szoftvercsere biztosított. Helyszín: TIT Stúdió, Budapest XI., Bocskay út 37.

A HELP és a SUPERMONITOR együttes működtetése

COMMODORE 64

A HELP beépített monitorja nem teljes értékű. Nem mutatja meg például leálláskor a tárolók (akkumulátor, X és Y indextárolók) tartalmát, és nincs „feltöltő" (fill) utasítása. A memóriában nehézkes az egyes programrészletek áthelyezése egy más területre, és ismeretlen a memóriakiíratás hexadecimális formában — az úgynevezett Memory Dump. De a legnagyobb hiányossága talán, hogy monitorprogramja nem teszi lehetővé rövid gépi kódú programok közvetlen beírását a memóriába: az *A*, az Assemble utasítás — a fordítóutasítás — teljesen hiányzik belőle, csak a visszafordítási lehetőség van meg; ez a *D* opció (*D*isassemble) teljes értékű, sőt igen jól kihasználható a folyamatos visszafordítás a szóköz nyomógomb lenyomásakor.

Az általam követett módszer szerint a HELP és a SUPERMONITOR program együtt működtethető, és mind a két program szolgáltatása egyidejűleg vehető igénybe. Először a HELP-et kell a C64-be betölteni, majd a SUPERMONITOR-t egyszerű BASIC programként. Ezután RUN-nal indítjuk el a SUPERMONITOR-t, aminek hatására átmásolja saját magát a HELP által kijelölt BASIC terület vége elé, és a BASIC terület végét jelző mutatókat átállítja; így BASIC programmal nem írható felül és nem rontható el sem a HELP, sem a SUPERMONITOR. X-szel kilépve a SUPERMONITOR-ból, NEW-val törölhetjük a BASIC területet (illetőleg, mint tudjuk, ez csak a mutatókat állítja vissza), ami által a BASIC számára mintegy 28 kbájt fog rendelkezésre állni.

```
8#
   PC   SR AC XR YR SP
.:0800 32 44 00 00 F6
.
.:002B 01 08 03 08 03 08 03 08
.:0033 ED 77 00 80 ED 77 82 FF
.:003B 00 00 00 00 00 00 00 08
.
.      TXTTAB VARTAB ARYTAB STREND
.:002B 01 08, 03 08, 03 08, 03 08,
.        1      2      3      4
.
.      FRETOP FRESPC MEMSIZ
.:0033 ED 77, 00 80, ED 77, 82 FF
.        5      6      7
```

Ha újra vissza akarunk lépni a SUPERMONITOR-ba, akkor azt a BREAK rutinon keresztül tehetjük meg: egyszerűen egy olyan bájtra ugratjuk a SYS(X) utasítással a BASIC interpretert, amely memóriarekesz 0-át tartalmaz. Ilyen a 2048-ik memóriatároló, ha a BASIC terület alját nem toljuk feljebb, és így SYS(2048) segítségével újra visszatérhetünk a SUPERMONITOR programba. Általánosabban megfogalmazva: a BASIC munkaterület kezdetének mutatói (a 43-44. tárolók) segítségével a SYS(PEEK(43)+256*PEEK(44)-1) formában mindig visszatérhetünk a SUPERMONITOR-ba, függetlenül attól, hogy a BASIC terület kezdete hol található.

Az elmondottakat illusztrálja az *ábra*, ahol a BASIC mutatók tartalmát nyomtattuk ki a SUPERMONITOR-ral, felhasználva a HELP-nek a nyomtatót megnyitó utasítását, a "(" jelet és a nyomtatót lezáró utasítását, a ")" jelet.

Felhívom a figyelmet arra, hogy a HELP monitorja saját céljaira — mégpedig arra, hogy az éppen aktuális kétbájtos tárcímet tárolja — használja az $FB és $FC tárolókat (251. és 252. tárolókat). Így azok a gépi kódú programjaink, amelyek ugyanezekkel a nullás lapon lévő — a BASIC interpreter által le nem foglalt — szabad tárolókkal dolgoznak, emiatt nem minden esetben szolgáltatják a helyes végeredményt, ha a gépi kódú programunk futtatását a HELP monitorja indítja el a * segítségével. Ilyenkor a hibás működés nem mindig írható a saját gépi kodú programunk rovására. Néha csupán arról van szó, hogy a HELP monitorprogramja „interferált" a mi gépi kódú programunkkal, mert módosította az $FB/FC tárolók tartalmát. Ezt a hibalehetőséget a SUPERMONITOR használata esetén nem tapasztaljuk, ami egy további érv a két program közös munkája mellett: a lefordított program tesztelését, futtatását a SUPERMONITOR programmal érdemes elvégeztetni.

A HELP működtetésével foglalkozik például Úry L.: Commodore 64. II. c. könyve (LSI ATSZ, Bp. 1984.), a SUPERMONITOR használata pedig megtalálható Lángos I.: A Commodore 64 mikrogép kezelése és programozása (Comporgan, Bp. 1984.) c. kiadványban.

SZABÓ PÉTER PÁL

A HELP után betöltött SUPERMONITOR program együttes helyfoglalása a memóriában, RUN és NEW után

1 - TXTTAB: $2B/2C - 43/44: $0801-re 2049-re mutat, BASIC szöveg elejének mutatója.

2 - VARTAB: $2D/2E - 45/46: $0803-ra, 2051-re mutat, a BASIC változók kezdetének mutatója.

3 - ARYTAB: $2F/30 - 47/48: $0803-ra, 2051-re mutat, a BASIC tömbök kezdetének mutatója.

4 - STREND: $31/32 - 49/50: $0803-ra, 2051-re mutat, ez a BASIC tömbök végének mutatója.

5 - FRETOP: $33/34 - 51/52: $77ED-re, 30701-re mutat, a szöveges változók területének az alját mutatja.

6 - FRESPC: $35/36 - 53/54: $8000-re, 32768-ra mutat, saját szöveges változók mutatója.

7 - MEMSIZ: $37/38 - 55/56: $77ED-re, 30701-re mutat, a BASIC terület végét jelöli ki.

Itt mutatták be a Multitech Plus 710-et

SOFTWARE '88

Kicsik és nagyok közös megméretése

1983-ban a Comporgan Rendszerházban született meg az ötlet, hogy a magyar szoftvergyártók és -forgalmazók rendszeresen – kétévenként – visszatérő kiállításon mutassák be termékeiket.
A SOFTWARE '88-at a Duna Inter·Continental szállodában rendezték meg.

A találkozón részt vett a magyar gyártók és forgalmazók színe-java. A 68 kiállító között megtalálhattuk a kutatóintézeteket és nagyvállalatokat, de az egy-két fős vállalkozásokat is. Folytatódott a számítástechnikai kisszövetkezetek előretörése. A bemutatott és eladásra kínált szoftverek többsége IBM és azzal kompatibilis gépekre készült. Úgy tűnik, hogy a Commodore és társai kiszorultak — a kezdeti fellendülés után — a vállalati alkalmazásokból.

A SOFTWARE '88 rendezvénnyel egy időben, de külön kamarakiállításon mutatták be a számítógéppel segített mérnöki tervezés és gyártás (CAD/CAM) hazai termékeit. A rendezők tizenegy hazai és külföldi céget hívtak meg, és ezen a kiállításon jelent meg először az erre a területre szakosodott magyar—osztrák—amerikai vegyesvállalat, a Flexys Gyártásautomatizálási Rt. A nagyok mellett — mint az MTA—SZTAKI, KFKI, Videoton — sikert aratott az Alkalmazástechnikai Kisszövetkezet. A MODBUILD 3 dimenziós, geometriai modellező programcsomag és a COOPGRADING—A számítógépes konfekcióipari szériázó és interaktív termékoptimalizáló rendszer bemutatója előtt állandósult a tolongás, ami egyébként mindkét kiállítás jellemzője volt.

A bemutatóval párhuzamosan — ez már hagyomány —, megrendezték a szoftvervásárt, amire 77 terméket neveztek. A bírálóbizottság — ami a számítástechnika alkalmazásában különösen érdekelt főhatóságok és szakmai cégek képviselőiből állt — négy terméket díjazott.

Közműtérkép számítógépen

Az első díjat (százezer forintot) a SOFT—COOP Kisszövetkezet GRATIS nevű grafikus adatbáziskezelő és -tervező interaktív szoftvere kapta. Az adatbáziskezelő és műszaki tervezőrendszer százötven grafikus funkciót valósít meg. A hierarchikus szervezésnek köszönhetően az akár több százezer vonalat tartalmazó kézi adatbázisból tetszőleges objektum néhány másodpercen belül azonosítható, manipulálható. Az egyes objektumokhoz szimbolikus, vagy részletesen ábrázolt képi megfeleltetések rendelhetők. Külön előnye, hogy az adatokhoz szükségszerűen kapcsolódó kiegészítő műszaki információ tárolása és feldolgozása is lehetséges. A műszaki tervező része teljes 2D és drótváz, illetve síklapmodell szintű 3D grafikus tervezési, dokumentálási, archiválási funkciókat lát el. A felhasználóbarát program nagymértékben elősegítheti a számítógépes tervezés itthoni elterjedését.

Programozott párválasztás

Érdekes felhasználói területre készült szoftver kapta a második díjat (hetvenötezer forintot): dr. Fischer Róbert, Danyi Zoltán és Bessenyei István COloMP nevű, a szarvasmarha-ágazat operatív irányítására alkalmas rendszere. A program lehetővé teszi a legkedvezőbb tenyésztési és tejelési módszer kialakítását, az állományváltozás tervezését, a takarmánykészletezés és optimalizálás megvalósítását. A rendszer számos előnyös tulajdonsággal rendelkezik:

- Az adatfelvitel szubjektív, nincs kötött kódrendszer. A felhasználó maga alakítja ki adatrendszerét.
- Több modulból áll, ezek egymástól függetlenül alkalmazhatók (pl. takarmányoptimalizálás).
- Kezelése nem igényel számítástechnikai tudást.
- Részletes listát készít az állategészségügyi eseményekről.
- Törzstenyésztő előrejelzéseket ad (pl. várható ellésről).
- Párosítási tervhez meghatározott szempontokat használ (pl. húsminőség, tejelés stb.).

A COloMP-ot több gazdaságban bevezették és sikeresen alkalmazzák.

Feltörhetetlen programok

A harmadik díjat (ötvenezer forintot) az Éltető László által készített EltGUARD másolásvédelmi és hierarchikus programvédelmi rendszer kapta. A program alapvetően két funkciót valósít meg: megakadályozza az illegális másolást és az illetéktelen személy által való programindítást. Ezáltal az adatállományt is védi. Legjobb referencia róla, hogy két éve használják és az EltGUARD-dal védett programot még senkinek nem sikerült feltörni. Külföldön is nagy az érdeklődés iránta, ezért többnyelvű DEMO-program és kézikönyv is segíti a felhasználók munkáját. Közel harminc cég használja programjai védelmére. Közöttük van a szakma színe minden „nagyja". Ezzel a rendszerrel védi például az Állami Népességnyilvántartó Hivatal is programjait, adatait.

A negyedik díjat az AMT által nevezett Kanyar nevű program nyerte (huszonötezer forintot). A terméket lapunk 1987/8. számában már ismertettük.

A kiállításon továbbra is az úgynevezett ügyviteli alkalmazási programcsomagok (bér-, munkaügyi, raktárnyilvántartási, statisztikai programok) domináltak, de — mint az a cikkből is kiderül —, széles volt a termelést segítő rendszerek köre is. A kiállítás utolsó napján rendezett szakmai kerekasztal-beszélgetésen elhangzott, hogy a magyar számítástechnikai termékek tőkés exportja 1986-ban elérte az ötszázhetvennyolcmillió forintot, ez az 1981-es évi exportnak a négyszerese.

— p —

Egy valóban oktató „játék”: a Domesday-rendszer

1086-ban, uralkodásának huszadik évében Hódító Vilmos angol király fel akarta méretni a tulajdonát képező birodalmat: országos összeírást rendelt el. Az összeírás egy évet vett igénybe, de egy napra vonatkoztatták, a „világ vége” napjára, angolul Domesdayre.

Nem lehet véletlen ez a sötét tónus: a Hódítónak nyilván végítéletszerűen kellett elfogadtatnia uralmát a leigázottakon, s a folyamatot tetézte a birtokosi „leltár”. Mindennek vége — minden a királyé —, gondolhatták az akkoriak. A felmérés eredményét a király számára az ún. Domesday Bookban foglalták össze.

Ennek az első nagy összeírásnak a 900 éves évfordulójára emlékezve, néhány brit vállalat összefogott, és kifejlesztett egy rendszert, ami lehetővé teszi a 80-as évek Nagy-Britanniájának is valami hasonló leltározását, majd az országnyi vagyon részletes bemutatását: az adatok, információk, képek sok szempontú lekérdezését. A feladatot Domesday Projectnek nevezték el. Az információk összegyűjtéséhez országos méretű kampányt szerveztek, amelyben vállalatok, szervezetek, helyi közösségek működtek közre, és összeállították a bemutatásra érdemesnek tartott dokumentumokat.

Az összefogással létrehozott berendezés és program, valamint a válogatott és rendezett adatokat tartalmazó videolemezek lettek a Domesday Project eredményei. A Domesday-rendszert elsősorban az iskolákban terjesztik, ahol az oktatás kiváló segédeszközeként hasznosulhat.

Angliai tartózkodásom során alkalmam nyílt megismerkedni ezzel a nagyon szellemes és érdekfeszítő rendszerrel.

A felhasználó és a rendszer közötti kommunikációt egy klaviatúra, egy kezelőgombot tartalmazó egér és egy képernyő teszi lehetővé. A kezelőgombbal a kurzor mozgatható tetszőleges irányban. A rendszerhez jelenleg két videolemez tartozik: az ún. nemzeti/országos (national) lemez és az ún. közösségi (community) lemez. Ezek bemutatását sajnos csak megkísérelni lehet — valójában mindenkinek a rendszer működése közben kellene látnia! . . .

Az első lemez a kultúrára, a társadalomra, a gazdaságra és a környezetre vonatkozó adatokat tartalmaz. A kívánt információt menük sorozatán keresztül érhetjük el. Ugyanazt az adatot közvetlenül is megkaphatjuk azonban kulcsszavak megadásával. A menüsorozaton végighaladva, a legalacsonyabb szintű menü lényegében egy tartalomjegyzék, amely megmutatja, hogy milyen adatokat lehet kiválasztani és ezeknek az adatoknak mi a típusuk: számadat, szöveg vagy kép.

A számadatok a menükkel kiválasztott témához tartozó statisztikai adatokat jelentenek, amelyek ábrázolási módja vonalas görbe, oszlopos vagy körcikkes lehet — hogy melyik, azt a felhasználó határozza meg. A szövegek a behatárolt tárgyról szóló újságcikkek, illetve -részletek. A képek fényképeket jelentenek.

A pillanatnyi állapotból (menüből) természetesen visszafelé is lehet haladni. Képzeljük el például, hogy milyen nagyszerű lehetőség, ha egy kirándulásra készülve a természeti szépségeket bemutató rész előzetes kiválasztása után megnézhetjük (és ki is nézhetjük magunknak) a turistaházakat — a szálláshelyek adatait: a szobaszámot, a komfortfokozatot, a szobák árát —, és képzeletben már előre bejárhatjuk úticélunk nevezetességeit, továbbá megtudhatjuk a bejárni kívánt útvonal hosszát.

A kulturális élet — mint almenü — kijelölésével képtárak, színházak, iskolák nevezetességeit tekinthetjük meg. Ugyanezekkel a képekkel az ún. Domesday Galery (galéria) egy-egy részében is találkozhatunk. Bármelyik menüből át lehet térni ugyanis a galériába, ami egy fiktív kisváros és a benne lévő fiktív kiállítási épület, amelyben az egér által lehet közlekedni. A képernyőn mindig az a kép jelenik meg, amit az egérrel megközelített helyen a látogató láthat — vagyis vannak áttekintő és közelképek, esetenként további részletek, amelyeket a felhasználó megnézhet, vagy elhalad mellette. Mindig meg kell adni azt az irányt, amelyen tovább akarunk menni, és ha ez lehetséges (mert nem lehet mindig feltétlenül minden irányban továbbjutni), akkor a következő kép az új helyen képződő látványt mutatja. Így „megbámulhatók" az egyes képek, át lehet vonulni egyik teremből a másikba, meg lehet nézni egészen közelről egy-egy kiválasztott kiállítási tárgyat (mondjuk Shakespeare szülőházában a konyhafelszerelés egyes eredeti darabjait) vagy egy festmény részletét. Ki lehet menni az utcára is, ott „séta" közben tanulmányozni a műemlék környezetét stb. Szükség esetén bármikor lehívható az épület alaprajza vagy a település térképe, hogy a haladási irányt pontosabban kiválaszthassuk.

Ezen a lemezen vannak továbbá a 80-as évek legjellemzőbb BBC-híradóinak részletei, amelyeket évenként csoportosítva nézhetünk végig.

A második, az ún. közösségi lemezre elsősorban a különböző csoportok, társaságok (iskolák, vállalatok, községek, városok, klubok stb.) által összegyűjtött és elkészített képeket, szövegeket, va-

lamint statisztikai adatokat vittek fel. A tartalom hat szinten keresztül érhető el. Az egyes szintek:

- 0-ás szint: az egész ország. Szatellit-felvételeket láttat és szöveges ismertetést nyújt.
- 1-es szint: országrészek. Szintén szatellit-felvételeket és leírást mutat.
- 2-es szint: 40 × 30 km-es területeket fog át. Szatellit- és légi felvételeket, a terület szöveges bemutatását, térképeket nyújt.
- 3-as szint: 4 × 3 km-es területek. A közösségek által beküldött fényképeket, dokumentumokat és térképeket foglalja magába.
- 4-es szint: utcatérképek. Ugyanaz, mint az előző, de csak néhány utcáról.
- 5-ös szint: alaprajzok. Nevezetesebb épületek alaprajzát, fényképeit és leírását adja.

Bármelyik szinten a kérdéses területre nézve van egy tartalomjegyzék, amelyből ki lehet választani a kellő fényképet, szöveget vagy térképet. A megfelelő szintű térképekre különböző körzethatárokat rajzolhatunk be: például közigazgatási határok, választókerület határai, elektromos, postai vagy egyházi körzetek határai stb. A vizsgált területről különböző statisztikák kérhetők, és ezeket a térképre ad-hoc csoportosításban lehet felrajzoltatni: például a kérdéses postai körzetben mekkora a 150 m²-nél nagyobb alapterületű és két telefonvonallal felszerelt lakások aránya. (Milyen könnyű az angol snoboknak, ha lakást akarnak választani! . . .) Ezután a postai körzet térképére ezeknek a lakásoknak az elosztását is fel lehet rajzoltatni: alapértelmezésben például 0—1,5 százalékig zöld, 1,5—4 százalékig piros, 4—10 százalékig sárga, 10 százalék fölött kék színnel, ami azt jelenti, hogy az előbbieknek megfelelően színezett térképet kapunk. A százalékértékeket meg is változtathatjuk, és akkor egy másképpen színezett ábránk lesz. Nem kell hangsúlyoznom, hogy micsoda kiváló lehetőség az összehasonlításra, elemzésre.

Használat közben térképről térképre haladunk: vagy ugyanabban a szintben, vagy eggyel nagyobb vagy kisebb szinten. Egy-egy térképet a hely nevének megadásával, a hely földrajzi koordi-

nátáival vagy pedig a kezelőgolyóval térképről térképre „gurulva" választhatunk ki; megannyi variáció a felhasználó ízlésének, képességének, kedvtelésének, konkrét és céltudatos munkájának segítésére.

Összefoglalva: az ismertetett két lemezzel a Domesday-rendszer átfogó és részletes képet nyújt évtizedünk Nagy-Britanniájáról, méghozzá élményszerűen: könnyedén, majdnem játszva — úgy, hogy még a legmegátalkodottabb kétkedőknek és legnevelhetetlenebb „sajtkukacoknak" is leköti a figyelmét. Nem véletlen tehát, hogy bizonyos szűkebb körökön kívül elsősorban az iskolákban, az oktatásban akarják terjeszteni, propagálni. Ez indokolta az egész projekt koncepcióját is: a videolemez csak nagyon jelentős példányszám mellett kifizetődő. A Domesday Project irányítói arra számítanak, hogy a közép- és felsőfokú iskolák mellett könyvtárak, a különböző szintű államigazgatási szervek, írók, újságírók, filmesek, televíziós vállalatok, kereskedelmi vállalatok, utazási irodák, ingatlanközvetítők és fuvarozó cégek is mindennapi munkaeszközként alkalmazzák majd a rendszert.

Mi még a reményeiket is csak irigyelhetjük.

DOBÓ CSABA

Az év számítógépei

Immár hagyomány, hogy az év számítógépét a nyugatnémet CHIP magazin kezdeményezésére kilenc ország szakújságíróiból álló zsűri választja ki. A kilenctagú zsűrinek az elmúlt évben két szocialista országbeli tagja is volt, a lengyel Komputer és a magyar Új Impulzus szerkesztősége munkatársainak személyében. Az elbírálásra kerülő gépeket kategorizálták, és a kategóriákon belül a szerkesztőségek 100 pontot oszthattak ki legfeljebb öt gép között.

A házi számítógépek kategóriájának nyertese a Commodore AMIGA 500-as lett 460 ponttal. Jellemző a győzelem mértékére, hogy a második helyezett 105 pontot, míg az utolsó, tizedik induló csak 15 pontot ért el.

Ismerkedjünk meg nagy vonalakban a győztessel. Az AMIGA 500 kifejlesztésével a Commodore cég a népszerű, több mint hétmillió példányszámban értékesített C64-es gépet kívánta kiváltani.

A gép mikroprocesszora a MOTOROLA által gyártott MC68000, ami 16 bites buszszal és 32 bites bitregiszterrel rendelkezik. A mikroprocesszort — az AMIGA „nagyobb" testvéreihez hasonlóan — három cooprocesszor segíti. Ez a három nővér „Fat Agnus" (Kövér Ágnes), „Denise" és „Paula" névre hallgat. Fat Agnus, ez a kövér „hölgy" látja el az AMIGA 500 egyik központi, ún. BLITTER funkcióját (BLITTER=BLOCK IMAGE TRANSFERRER), ami tetszőleges mennyiségű képpontcsoport gyors mozgatását jelenti egyik képernyőpozícióból a másikba. Így válik lehetővé, hogy másodpercenként egymillió képpont jelenhessen meg a képernyőn, nagy sebességű animálást megvalósítva. A dinamikus RAM időbeli vezérlési feladatainak egy részét is ez az áramkör végzi. Ezenkívül a közvetlen tároló-hozzáférést (DMA) is koordinálja. Ebben a chipben található még az ún. COPPER is, ami a grafikus vezérlőregiszter-tartalmakat frissíti fel az elektronsugár képernyőn történő függőleges visszafutása alatt.

Denise a képernyőformátumok vezérlését végzi,valamint a 4096 féle színt tartalmazó palettából a szín-hozzárendelési feladatokat is ellátja. Ez az áramkör tartalmazza a szövegvezérlést (60 vagy 80 karakter soronként), valamint felelős a sprite funkcióért is. A sprite-ok kicsi, gyorsan mozgatható képernyőalakzatok, melyeket Denise nyolc sprite-DMA csatorna segítségével generál. Ebben az áramkörben található továbbá a géphez csatlakoztatható egérlogika és a képernyőn szabadon mozgatható alakzatok ütközésfigyelése is.

Paula a jó sztereo hanghatásért felelős. Két különálló csatornán négyhangú hanghatást is képes biztosítani. A frekvenciaterjedelem kilenc oktáv. Frekvencia- vagy amplitúdómodulációs üzemmódokra alkalmas. Paula végzi a kétirányú portok, valamint a készülék házába épített floppymeghajtó vezérlését is.

Végezetül szólnunk kell néhány szót a házi számítógépek világában forradalminak számító operációs rendszerről, az AMIGA-DOS-ról. Ez egy multiprogramozású operációs rendszer, amely a felhasználó számára kívülről nézve teljesen „átlátszó". Úgy tűnik, mintha minden, a számítógéprendszerben futó programnak saját mikroprocesszora lenne. Úgy is mondhatjuk, hogy minden program „megkapja" a saját, virtuális mikroprocesszorát. Mivel az AMIGA 500-ban egyetlen 68000-es mikroprocesszor szimulálja a virtuális processzorokat, az operációs rendszer minden egyes taskhoz (feladathoz) prioritásának megfelelő időszeletet rendel (time-slicing). Ha a task túllépi a számára kijelölt időtartamot, akkor az AMIGA-DOS megszakítja annak futását, és egy másik ugyanolyan prioritású task kap futási lehetőséget (round robin módszer). Az AMIGA-DOS 256 prioritási fokozattal rendelkezik.

SZ. S.

A díj és az AMIGA 500

Műszaki adatok

Mikroprocesszor:
80C88: 4,77/8 MHz

Munkatároló:
640 kbájt RAM (1,6 Mbájtig bővíthető)

Háttértároló:
3,5 collos floppy, 720 kbájt kapacitás, 10 Mbájtos fix lemez (nem winchester!)

Képernyő:
Szuper sarkított kristályos folyadékkijelző (háttérvilágítással) 26,5 cm átmérő

Klaviatúra:
10 funkcióbillentyű, különálló kurzormozgató blokk, elkülönített numerikus billentyűzet

Interfészek:
Soros (RS232), párhuzamos (Centronics), RGB Video, külső floppy-, ill. fixlemezmeghajtó

Súly: 7 kg

A kategóriagyőztesek

Kategória	*A gép neve*
PC8086/8088	IBM PS/2 30-as modell
PC80286/80386	TANDON PAC 286
PC68000/68020	APPLE MACINTOSH II
Hordozható	COMPAQ PORTABLE III
Kézben tartható	ZENITH Z183

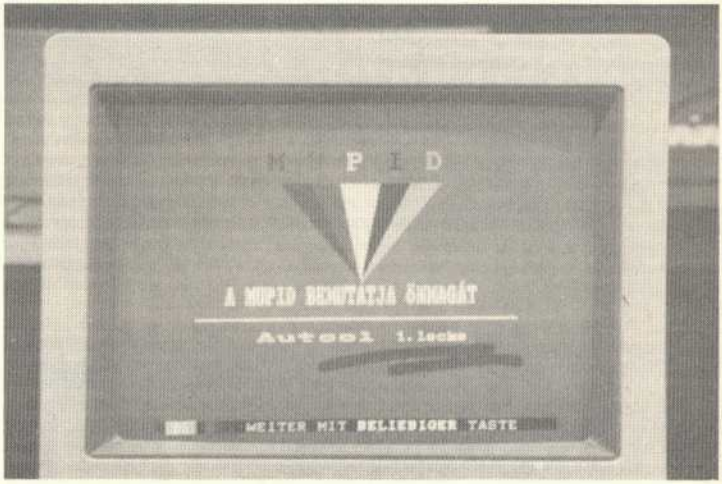
Miért nem stupid a MUPID? avagy: Milyen az intelligens videotex-dekóder? V.

Sorozatunkban a MUPID intelligens vtx-dekóder lényegesebb jellemzőit és főbb alkalmazásait ismertettük. Zárásként egy különleges területét, a vtx-rendszeren is terjeszthető ismeretek átadásában, átvételében való egyedülálló szerepét vázoljuk.

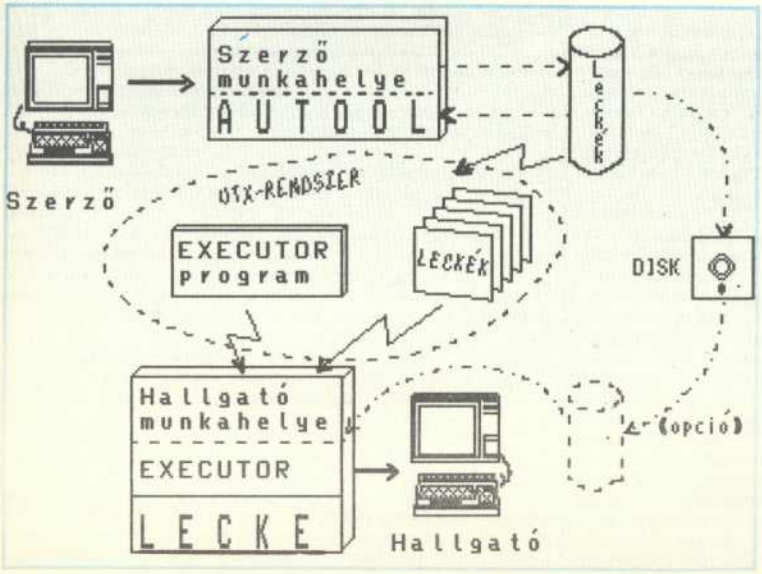
MUPID az oktatásban

A MUPID — duális funkciói révén — a távoktatásban is jelentős szerepet játszik. Az amerikai Control Data cég PLATO fantázianevű oktatórendszerét bővített formában adaptálták a MUPID-ra. Az AUTOOL néven ismert ún. szerzői rendszer lehetőséget nyújt számítógépen futó leckék írására a legkülönbözőbb témákban. A szerzői rendszer kifejlesztésekor ügyeltek arra, hogy alkalmazása országos hálózaton — például videotexen — is elérhető legyen.

Az AUTOOL leckék több videotexszerű képből állnak, amelyeket a szerző MUPID vtx-dekóderrel, M—DISK-kel és szerkesztőprogrammal (AUTOOL Editor) készít el. A program alkalmazói felülete olyan, hogy az egyes oldalakra — minden számítástechnikai ismeretet nélkülözve — menüválasztásos technikával, a MUPID billentyűzetéről vihetők be a vtx megjelenítési jellemzőkkel rendelkező szöveges, grafikus információk, animációs lehetőségek és a hallgató által megválaszolandó kérdések. A szerkesztés eredménye a képernyőn egyidejűleg meg is jelenik. A szerző egy különleges tesztelő programmal (Test Executor) a leckéket a hallgató szemszögéből áttekintve, azokat vizuális és didaktikai szempontból ellenőrizheti. A csak MUPID-dal rendelkező hallgatók a vtx-központon keresztül, azok pedig, akiknek M—DISK-jük van, floppyn is hozzáférhetnek a leckék futtatásához szükséges Executor programhoz — ez nyújtja a megjelenítést, értékeli a hallgatói válaszokat és vezérli az egész lecke lefutását —, valamint az egyes leckékhez (lásd az *1. ábrát*).



1. ábra

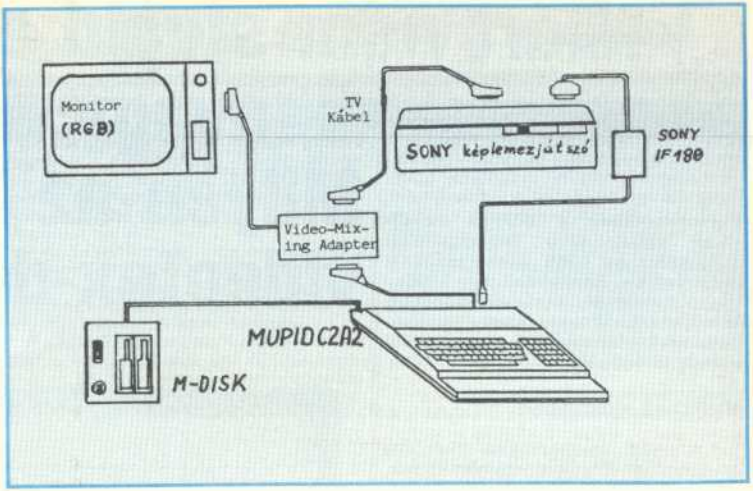


1987 őszén mintegy 100 lecke volt az osztrák vtx-központban. A témák változatosságára mi sem jellemzőbb, mint hogy a MUPID kezelését oktató leckétől kezdve az AIDS-ről szóló felvilágosító és különböző nyelvoktató leckéken keresztül a szörfözés rejtelmeit bemutató kurzusokig igen széles a skála. A MUPID alkalmazhatóságát a hazai oktatásban a szakemberek vizsgálják. (Erről remélhetőleg a következő számban számolnak be az illetékesek. — A szerk.)

Szintén az oktatás területén nyújt támogatást elsősorban, de felhasználható különböző célú demonstrációkra is a MUPID-nak egy olyan hardver és szoftver konfigurációja, amely lehetővé teszi képlemezjátszó vagy képmagnó vezérlését, a videoanyag feliratozását, azon változatos magyarázó grafikák elhelyezését, a sztereo hangcsatornák egymástól független működtetését (nyelvtanulásnál) és a felvételek visszajátszását különböző sebességekkel.

Minderre azért van lehetőség, mert a CEPT-ajánlás megengedi a videokép vtx-képen való megjelenítését úgy, hogy a videoképsík a vtx-képsík mögött helyezkedik el (vö. a képújság kevert megjelenítési módjával). A MUPID és egy Sony képlemezjátszó együttműködését megvalósító konfigurációt mutat be a *2. ábra*. A rendszer megfelelő működtetéséhez szükség van a képlemezjátszót vezérlő — parancsértelmezőt tartalmazó és a parancsok megszerkesztését támogató — programcsomagra, amelyet floppyról kell a MUPID-ba tölteni. A rendszerrel szerzett hazai tapasztalatokról még nincs tudomásunk.

A MUPID és lehetséges alkalmazásainak részletesebb bemutatása már csak terjedelmi okokból sem lehetett célunk. Sorozatunkban megpróbáltuk röviden vázolni sajátosságait és a dualitásából fakadó speciális alkalmazási lehetőségeket. Az ezeket összefoglaló táblázat látható a *3. ábrán*.



2. ábra

A MUPID szoftverrel való ellátottsága jónak mondható. A gyártó cégen kívül Maurer professzor intézete és több osztrák, valamint nyugatnémet szoftverterjesztő kínál speciálisan a MUPID-ra szabott programokat, mind az osztrák vtx-központból lehívhatóan, mind pedig floppyn. A MUPID koncepciót kiterjesztették opcióként az IBM PC-kre is két MUPID-kártya és egy szoftvercsomag hozzáadásával — de ez már egy más világ . . .

JURENKA OSZKÁR

3. ábra

M U P I D alkalmazások - HW/SW konfigurációk

* .. feltétlenül szükséges
Jelöletlen .. nem szükséges, de lehetséges

Megjegyzés:
minden konfiguráció SCART csatlakozós RGB monitorral értendő.
A vtx-terminált megfelelő, 75/1200 bps, vagy
1200/1200 engedélyezett modemmel lehet a
távbeszélő-hálózathoz csatlakoztatni.
Bármely konfigurációhoz csatlakoztatható nyomtató,
megfelelő nyomtatóprogrammal

ALKALMAZÁS	SZÜKSÉGES HW/SW	Standard MUPID	CMOS - RAM bővítő	Digitalizáló+kamera	M-DISK	Grafikustábla	Video-keverőadapter	Komfortos graf.szerk	AUTOOL - floppy	Digitalizáló floppy	Képlemezjátszó SW	CP/M op. rendszer	WordStar floppy	MULTIPLAN floppy	dBase II. floppy	M80 Assembler+Linker	Turbo-Pascal floppy
vtx-oldalak, TSW, oktatóprogramok, digitalizált képek lehívása		*															
vonali (online) szerkesztés és központba betöltés		*															
programfejlesztés és központba betöltés		*															
teleprogramok és vtx-oldalak tárolása RAM-DISK-ben		*	*														
standard helyi grafikus szerkesztés és programozás		*			*												
komfortos helyi grafikus szerkesztés grafikus tábla nélkül		*			*			*									
komfortos helyi szerkesztés grafikustáblával		*	*		*	*		*									
AUTOOL leckék előállítása		*			*				*								
digitális képek előállítása		*		*	*		*			*							
képlemezjátszó vezérlése		*			*		*				*						
szövegszerkesztés WordStar-ral CP/M alatt		*			*							*	*				
táblázatkezelés CP/M alatt		*			*							*		*			
adatbáziskezelés CP/M alatt		*			*							*			*		
CP/M fejlesztőrendszer Z80 Assembler programmokhoz		*			*							*	*			*	
CP/M Turbo-Pascal Compiler		*			*							*					*

● Csodálatos volt a reggel. Ren ilyenkor érezte a legjobban magát. A nap arany fénye beragyogta a vidéket, a meredeken az ég felé törő hegyek lejtőin lecsúszott a völgyekbe, hogy a zöld növények leveleit maga felé fordítsa. Felhangzott a rovarok nyugtató zaja, és feltámadt a szél, mint mindig kora reggel.

Ren elnyúlt egy fa tövében. Nem csinált semmit, nem gondolkodott semmin, csak élvezte a testében szétáradó meleget. Bámulta az eget, és egyszerre megmerevedett. Egy fekete valami zuhant az égből a föld felé. A zuhanó tárgy eltűnt a domb mögött, majd hatalmas robaj rázta meg a környéket. A föld is megremegett. Ren legyőzte rémületét és elindult az ismeretlen valami felé.

Felérve a dombra, sokáig méregette a kockát.

— Talán egy kő az égből, mint a múltkor — gondolta. Lement a kockához, aminek majdnem a fele a talaj szintje alatt volt. Körbejárta, de megérinteni nem merte, mert érezte, hogy még nagyon meleg. Képzeletben összehasonlította a múltkori égi kővel. Ez sokkal szabályosabb volt. De valakinek ilyenné kellett faragnia!

Úgy érezte, jelentenie kell a városban. Otthagyta, a település felé vette útját. — Mi lehet a célja? — töprengett magában az úton. — Minden valamiért van. Ez miért?

— Ren! Ren! — riasztotta meg egy kiáltás. A városban is hallották a becsapódást, és érezték a rengést.

Ren megszemlélte a közeledő csoportot. — Igen — futott át rajta —, Taj vezeti őket. Lecsap erre is. Ehhez is csak ő ért.

— Onnan jössz? — kiabált Taj már messziről.

Ren nem válaszolt mindjárt, megvárta, amíg közelebb érnek, és csak akkor mormolt egy igent.

— Mit gondolsz róla? — folytatta a kérdezősködést Taj.

Rent meglepte a vezető kíváncsisága. — Szerintem — kezdte —, mesterséges valami.

— Ne hamarkodd el! — hűtötte le egy pillanat alatt Taj. — Emlékezz a gömbölyű kövekre, azokat is találtuk, és senki nem csiszolta.

— De szögletes formát csak mesterségesen lehet csinálni! — próbálta Ren védeni a gondolatát.

— Majd megvizsgáljuk — zárta le a beszélgetést Taj.

A csoport folytatta útját a gép felé. Ren csatlakozott hozzájuk. — Itt, erre, a domb mögött — kalauzolta az újonnan érkezetteket.

Lementek a kockához.

— Most pedig megvizsgáljuk, hogy van-e benne élet — mondta színpadiasan Taj, majd hirtelen elordította magát. — Hé, van ott benn valaki?

— Ez őrült — gondolta hitetlenkedve Ren.

— Van ott benn valaki? — ismételte meg Taj.

A kocka némán állt, csúcsával a talajba fúródva. Ren megvonaglott. — Ez megőrült — gondolta, de nem mert szólni.

— Tapasztalhattuk, hogy ez a valami egy szikla, egy nagy darab kő — fordult Taj a többiekhez. — Hogy önmagában értelmes, azt már akkor sem gondoltam, amikor megláttam. De ha benne lett volna valaki, az biztos, hogy a zuhanásnál meghalt. Tanúim vagytok! Menjünk vissza és mondjuk el a többieknek.

A Gép minden érzékelőjét a jövevényekre szegezte. A hangrezgéseket elraktározta későbbi elemzésre.

Megvárta, míg eltávolodnak tőle. Kiadta az utasítást az egyik karbantartó robotnak, hogy tegye szabaddá a kijáratot. A Gép kinyitotta az ajtót. A robot az ajtóhoz gördült, és ásni kezdett. Egyre több földet kotort a kockába. A Gép leállította. Előkereste a nemrég talált programot, és utasította az energiaágyút hordozó műholdat, hogy lőjön a kocka mellé, az általa kiszámított koordinátákba.

Az energianyaláb óriási robajjal csapódott be. A földdarabok szanaszét repültek, és a kocka méltóságteljesen billent vissza az alapjára. A rögzítetlen robot hangos csattanással vágódott az egyik lézertároló fémlemezéhez. A Gép megszüntette a tartalékrobot stabilizálását, az a sérült szerkezethez gördült és javítani kezdte.

Ren korán reggel elhagyta éjszakai pihenőhelyét. Érdekelte a szikla — ahogy elnevezte magában. A nap aranyló csíkja jelezte még csak a hajnalt, amikor Ren már úton volt.

Már a dombról észrevette, hogy a szikla nem úgy áll, mint előző nap. Megrémült, de valami ismeretlen erő, melyet idáig még nem érzett magában, szinte tolta előre. Odaérve körbejárta a kockát. Mikor a második körbe fogott volna, a Gép kitárta előtte az ajtót.

Ren megdermedt. Elhátrált, és pár méterről bámult befelé a sötét nyíláson. Összeszedte minden erejét, és elindult az ismeretlen felé, az új felé. Belépett a kockába, és várt, vizsgálgatta a számára teljesen idegen műszereket. — Mit keresek én itt? — kérdezte magától és kifelé indult.

— Hé, van ott benn valaki? — a Gép igyekezett hiba nélkül visszaadni a mondatot.

Ren megtorpant. Nem értett semmit.

— Hé, van ott benn valaki? — ismételte meg a Gép Taj tegnapi mondatát.

— Te beszélsz? — kérdezte elképedve Ren. — Mozogtál is, hiszen tegnap nem így álltál itt. Te akkor értelmes vagy. Ugye?

A Gép hallgatott.

Rent elöntötte a keserűség, a meg nem értettség, és kifakadt.

— Nem értheted — kezdte —, de a mi népünknek csak nagyon kis része él már. Sokan, nagyon sokan elpusztultak a háborúban. Abban a háborúban, ami után két évvel lettem én. — Várt egy kicsit. — Tudod — folytatta —, számunkra a legfelsőbb parancs, hogy ne pusztuljunk el. A háborúban ezt nem lehetett teljesíteni. Félek a pusztulástól, de még nem láttam soha, soha. Akkor miért félek?

Elhallgatott. Néma csend telepedett a kockára. — Úgy vélem — mondta hirtelen —, hogy te nem vagy mesterséges. — Ren kántálni kezdett egy rövid mondatot. — Nem pusztulhatsz, nem a sajátod vagy! — ezt ismételgette váltakozó hangsúlyozással.

Utoljára hosszan elnyújtva, nagyon mély hangon. A Gépben felidéződtek az utolsó földi napok.

— Miért? — kérdezte magától. — Mi az, ami felidézte bennem a pusztulás képeit?

„Megtalálni azt, ami közös, ami mindenhol, mindenkinek ugyanazt jelenti" — visszhangoztak benne a Mester szavai.

Ren időközben elhallgatott, és némán ült az egyik szerelőroboton.

— Jó, hogy nem érted — szólalt meg hirtelen —, te még nem vehettél részt egy ilyen pusztulásban. Igaz, én sem. És mégis félek. — Felállt és a kijárat felé indult. — Holnap visszajövök — szólt vissza.

— Igen — nyugtázta a Gép.

Ren megtorpant és visszanézett, majd továbbindult a város felé.

Reggelente feltámadt a szél, mint mindig. A napok csendesen folytak el. A városban már napirendre tértek a nagy kő felett. Ren nem. Nap mint nap elment a Géphez, és ott próbálta meg tisztázni önmaga gondolatait. Csakhamar rádöbbent, hogy ennek a valaminek szüksége van arra, hogy beszéljen hozzá. Beszélt is órákon keresztül, fecsegett, számolt, elmesélte, hogy mi történt vele aznap. A Gép hallgatott, és rögzített minden hangot. Csak amikor érezte, hogy Ren elcsügged, akkor próbálkozott meg egy-egy tömondat kiejtésével, hogy a lényt további beszédre bírja.

Éjszakánként elemezte az aznapi szöveget, és lassan kialakult benne egy új nyelv, René.

A Nap ugyanúgy kelt fel, mint máskor. Rent ismét az úton találta. A kocka felé igyekezett. Az úton már minden göröngyöt, fűszálat ismert. Felvidította a kapcsolat, valami, ami az övé, egyedül az övé, bár ez még nem tudatosult benne.

Belépett a kockába.

— Már nem tudok mit mondani.

— Ennyi már elég volt — válaszolt a Gép. Úgy ítélte meg, elérkezett az idő, hogy hozzáfogjon a feladat megvalósításához. Kivetítette a Föld képét, és beszélni kezdett.

— Erről a bolygóról jöttem — kezdte. Földnek nevezték, és már nagyon régen nem létezik. — Hirtelen rádöbbent, hogy milyen lassan képes beszélni. Számára végtelennek tűnő ideig, tizedmásodpercekig kereste Ren nyelvén a megfelelő kifejezéseket. — Mesterséges dolog vagyok. Egy gép. Típusmegjelölésem WON—Y—2. Az első gép építés közben elpusztult. Az a célom, hogy a Föld lakóinak nyomtalan eltűnését meggátoljam. — A Föld képe elporladt, és helyét egy szinte áttekinthetetlen szövevény foglalta el. Így nézek ki belül. — A kép egyik szelete vörös színűre változott. — Ez a központi vezérlőegységem. Semmilyen mozgó alkatrésszel nem rendelkezem. Az információk továbbítását fény bonyolítja. Én vagyok a Föld. Tárolóimban megtalálható minden, ami jellemezheti ezt az eltűnt kultúrát. Feladatom, hogy értelmes lényeknek mutassam be és adjam át mindazt a tudást, amit építőim felhalmoztak. — A Gép befejezte a bevezetőjét. — Kérdezhetsz.

Ren sokáig nem jutott szóhoz. Az új fogalmak és azok a szóösszetételek, amelyeket még nem hallott, teljesen elkábították. — Te gép vagy? — nyögte ki nagy sokára. Mesterséges?

— Igen. Mire vagy még kíváncsi?

— Mindenre, mindenre!

— Pontosabban kell kérdezned.

— Rendben van. Honnan tudod, hogy az a bolygó, ahonnan jöttél, nincsen már?

— Amikor felrobbant, akkor vágódtam ki a világűrbe.

— Miért robbant fel?

— Mert építőim felrobbantották, de további részleteket nem tudok mondani, mert töröltem.

— Értelmetlen — állapította meg Ren. — Biztos háború volt.

— Igen. Mondd, hogyan nevezitek a bolygótokat?

— Nincs neve, hogy emlékezzünk a borzalmakra.

— Nevezzétek el újra. Egyszerűbb, mint állandóan körülírni! — A Gép továbbfűzte a gondolatot. — Legyen Föld. Nem . . . Újföld.

— De a Föld elpusztult egy háborúban. Nem?

— Azért „új", itt már nem lehet több háború.

— Jó — hagyta rá Ren —, majd előterjesztem a városban. Most megyek, hogy elmondjak mindent rólad!

A Gépnek Ren távozása után elég ideje maradt, hogy a Mester egy régebbi szöveganalizáló programja segítségével a „nyelvtanulás" alatti monológokból kiszedje a lényeget.

— Ki vagy mi volt az ellenség? — tette fel magának a kérdést a Gép. Ismételten átfutotta a szöveget. — A vízben éltek. Ha valóban léteztek, akkor kellett valami nyomot hagyniuk. — Újraelemezte a bolygó felszínéről készült műholdképeket. Az egyik öbölben talált egy várost. Jobban mondva az alapjait, de valószínűnek tartotta, hogy mesterséges építményegyüttesről van szó. Körülbelül száz kilométerre lehetett tőle. Magához rendelte a szerelőrobotját. Az engedelmesen odagördült a Gép központi egységéhez, és hatvannégy parányi rubinrúdját a csatlakozóba illesztette. Átáramlottak a koordináták, a feladat és az útvonal.

Miután a Gép befejezte, a robot kigördült a kockából és elindult a víz alatti romok felé. Elkerülte a várost: az itt lakóknak ehhez semmi közük nem volt.

Ren másnap jóval a megszokott idő után érkezett.

— Képzeld — mondta a kockába lépve — a tanács elfogadta. A bolygó neve ezentúl Újföld. Elmeséltem rólad mindent, hogy gép vagy, és meg akarod tanítani nekünk azt, amit az építőid már ismertek. Olyan ismeretekre tehetünk szert, amit el sem tudnánk képzelni! Két értelem összefonódása!

— Ren, mi az, hogy Tadx? — vágott közbe a Gép.

— A katasztrófára gondolsz? Tadx-ot fegyújtották a benne lakók. De van egy párja, Biy, az is város volt. Egy szélviharban pusztult el. A háború Tadx-i, mert értelmetlen volt.

— Ezt ti állapítottátok meg — szögezte le a Gép.

— Mesélj az emberekről! — kérte Ren.

A gép válasz helyett egy filmet kezdett vetíteni. Ren némán figyelte a semmiből előbukkanó alakokat. A Gép eközben a robotról érkező hiradásokat gyűjtötte és elemezte.

— Van neked valamid? Ami a tiéd, csak és kizárólag? — kérdezett utána a Gép.

— Nem, nincs.

— Miért?

— Nem kell.

— Egyformák akartok lenni? Láttad a filmet. Az emberek testét ruha borította. Tudom, neked nem kell, de a ruha sok mindent elmond arról, aki viseli. Például a belső világát, egyes egyedek összetartozását.

— Mi mindannyian összetartozunk.

— Igen, de biztosan lesznek olyanok, akik észreveszik majd a kőbe faragott képed, ők egy kicsit közelebb állnak hozzád.

— Lehet. Ez új. Gondolkodnom kell.

— És a vallásotok. Mit tisztel?

— Egy eszmét, egy parancsolatot.

— Hogy néz ki?

— Sehogy — válaszolt elképedve Ren. — Nekünk nincs szükségünk arra, hogy egy idegen testbe bújtassuk a szellemet. A parancsolatunk a pusztításellenesség.

— A béke — szögezte le a Gép.

— Sok, túl sok mindent mondasz, helyükre kell tennem a dolgokat! — Ren felállt, és vissza sem szólva elindult kifelé.

A Gép értetlenül állt Ren távozása előtt. Megpróbálta szimulálni a várható viselkedésreakciókat, de a távozásnak mindössze huszonöt százalék esélyt adott. Felmerült benne a Mester egy régi mondása:

„Te mindig gép maradsz. Vannak dolgok, amiket sohasem tudsz majd értelmezni. Mindenben logikát keresel, és amiben nincs, azt törlöd. Te megteheted, hiszen gép vagy, és ezért maradsz mindig csak gép."

Jelzést kaptt a robottól, hogy értékelhető információkat talált. Haladéktalanul visszarendelte.

Besötétedett. Renben összevissza kavarogtak a gondolatok. Nem szeretett elmenni a Géptől, de most valami különös erő szinte taszította kifelé. Elhagyta a sík vidéket, már csak egy kis erdőcske választotta el a dombtól, amiről már látszott a város. Úgy tünt neki, mintha alak vált volna el a fák sötétjétől. Lassított, és nagyon megrémült. — Menekülni! Az ellenség! — vágott belé. — Menekülni! — dobolt az agyában. Meggyorsította lépteit, de az árny elállta az útját.

Képernyőmásolás

Az itt közölt programok C64 számítógépre érvényesek.
A listák és ábrák Citizen 120D típusú nyomtatóval készültek.
A futási idők is erre a párosításra vonatkoznak.
A tárcímek és a képernyővel kapcsolatos adatok
átírása után a programok a rokongépeken is futtathatók.
A nyomtatóra nincsenek különleges kikötések.

A képernyőmásolás (hardcopy, hardprint) egyszerű és sokoldalú eljárás futtatási eredmények gyors rögzítésére, képernyőelrendezések, felsorolásos adatok — különösen táblázatok — nyomtatására. Alkalmazásával sok programsort és programozói munkát takaríthatunk meg, ha van a célnak megfelelő, praktikus másolórutinunk. A programozók többsége őriz a gyűjteményében a karakteres képernyő kinyomtatására készült rutinokat, és az elterjedt segédprogramok egy része is tartalmaz ilyeneket.

A baj az, hogy e rutinok nagy része hadilábon áll az inverz karakterekkel vagy az idézőjelekkel, és különösképpen ezek kombinációival. A teljesen vagy majdnem hibátlanul másoló rutinok pedig nem paraméterezhetők, tehát nem rugalmasak, gyakran nem illeszthetők az adott feladathoz.

Íme néhány példa. A Commodore nyomtatók kezelési könyvében található egy — ránézésre is ijesztő — hardcopy rutin. Később egy nyomtatási képpel bemutatjuk, hogy mit tud és mit nem. Egyéb hibái mellett sajnos elfogadhatatlanul lassú is.

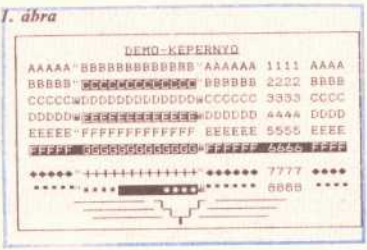
A PRINTER BASIC rutinja egyetlen az általam ismertek közül, amelyik minden karakterkonfigurációt abszolút korrekten másol, viszont egyetlen paraméterrel sem befolyásolható, és természetesen nem építhető be a saját programjainkba szubrutinként.

A Simon's BASIC-kel hasonló a helyzet, de ez ráadásul nem ismeri az inverz idézőjelet. Továbbá mindkettővel van még egy közös probléma, de erről majd később.

Foglaljuk össze hát, hogy milyen követelményeket kell támasztanunk egy olyan képernyőmásoló programmal szemben, amellyel valóban eredményesen dolgozhatunk, és amelyik kiterjeszthető ennek a hasznos eljárásnak az alkalmazási lehetőségeit:

— hibátlanul másoljon bármilyen karakterláncot,
— legyen BASIC-ben is elfogadhatóan gyors működésű,
— szubrutinként lehessen használni,
— a nyomtatási kép automatikusan kövesse a számítógép grafikus, illetve szöveges üzemmódját,

1. ábra



— a nyomtatási kép legyen tabulálható,
— ne csak a teljes képernyőt, hanem annak egy tetszőleges részletét is lehessen nyomtatni,
— a nyomtató soremelése letiltható legyen ott, ahol a képernyőn üres sorok vannak,
— több képernyőmásolást folyamatos nyomtatási képpé lehessen összefűzni.

A megoldás gondjai

Miután a fenti követelményeknek megfelelő rutint nem találtam és az irodalomban is hiába kerestem valami támpont után, fel akartam deríteni, hogy mi lehet a buktatója a dolognak. Hiszen az alapfeladat nagyon egyszerű: a képernyőtárból kiolvasott

2. ábra

kódokat ASCII kódokká kell alakítani és a nekik megfelelő karaktereket kinyomtatni.

Nos, kiderült, hogy a megoldás addig egyszerű, amíg nem akarunk idézőjelben inverz szöveget vagy éppenséggel inverz idézőjelet kinyomtatni, illetve megelégszünk a képernyő inverz karaktereinek normál karakterként történő nyomtatásával. Egy ilyen célra készült rutin is hasznos, mert vele a gyakorlatban adódó feladatok nagy részét meg lehet oldani, és igazán tömören megfogalmazható.

Az idézőjeles inverz szövegek másolásával az a gond, hogy a programban együtt kell használni a PRINT# utasítást, a CHR$(34)-et, magát a kiírandó idézőjelet és a CHR$(18), valamint a CHR$(146) inverzvezérlő karaktereket. Ezeket pedig olyan belső utasítások kapcsolják össze egymással, amelyek lehetetlenné teszik felhasználásukat mindazon kombinációkban, amelyekre szükségünk lenne. Ezt próbálta kijátszani a gyári program meglehetősen körmönfont módon, de sikertelenül.

Sok próbálkozás után úgy láttam, hogy nincs más megoldás, mint a CHR$(34) által definiált idézőjel helyettesítése valami mással, a legegyszerűbben a CHR$(39)-nek megfelelő felső vesszővel. Ez ugyanolyan tulajdonságú karakter, mint az összes többi, így egy egyszerű rutinnal megoldható az inverz karakterek nyomtatása is.

Az elegáns megoldás viszont az, ha a kettős idézőjel normál és inverz változatának sztringjét magunk állítjuk elő, és ezeket a nyomtató „bit image" üzemmódjában visszük papírra. A nyomtató üzemmódváltása önmagában nem gond, de ha a komplex feladat legkézenfekvőbbnek tűnő programozási megoldását választjuk, akkor csak tabulálatlan állapotban lesz a nyomtatási kép minden esetben hibátlan! Tabulált állapotban bizonyos karakterláncok a sorvégekről átkerülnek a következő sor elé. (Lehet, hogy ezért nem találtam egyébként kifogástalanul működő rutint tabulálható változatban?)

Ezt az akadályt végül is sikerült elhárítani, de ez részben összefügg egy másik — most már utolsó — probléma megoldásá-

val. A nyomtatónak minden egyes inverz karakter után készen kell állnia arra, hogy a következő karaktert esetleg normál kivitelben nyomtassa. A program oldaláról ennek az a legegyszerűbb megoldása, hogy az inverzben írandó karaktereket egyenként egy INVERZ BE és egy INVERZ KI vezérlőjel közé zárva küldjük a nyomtatóra. Erre viszont éppen a drágább, gyorsabb és az inverz karaktereket szebben író nyomtatók reagálnak azzal, hogy minden egyes inverz jel kiírása után egy karakterhellyel visszalép a nyomtatófej, majd innen halad ismét tovább. Ez egyrészt kellemetlen, rángatódzó fejmozgást eredményez, másrészt rendkívül lelassítja a nyomtatást. Sajnos a PRINTER BASIC is és a Simon's BASIC is ilyen működést vált ki; erre már utaltam.

A másolórutinok tesztelésére készítettem egy demonstrációs képernyőt *(1. ábra)*. Ez tartalmazza mindazokat a kritikus karakterláncokat, amelyek nyomtatási képe alapján egy rutint minősíteni lehet. Hasonló sorokkal ki-ki ellenőrizheti a birtokában lévő rutinok képességeit. A tabulálhatókat tabulálva is ki kell próbálni! Az ajánlott gyári rutin a *2. ábra* szerint nyomtatja ezt a képernyőt. A sortávolságot itt is és a későbbiekben is 1/8 collra állítottuk.

Képernyőmásoló rutinok

A következőkben az inverz nyomtatásra is alkalmas rutinokkal együtt összesen három programlistát mutatunk be. Ezekben az azonos funkciójú sorok mindig ugyanazt a sorszámot kapták. Az áttekinthetőség kedvéért a listák a szükségesnél jobban fel vannak bontva, valójában mindegyik kevesebb sorban leírható. Felépítésük olyan, hogy leggyorsabban a betűket dolgozzák fel, ezt követően a számokat, majd a grafikus és végül az inverz karaktereket.

A gyorsaság az ilyen jellegű pogramoknak egyébként is kulcskérdése; ennek érdekében igyekeztem kihasználni mindazokat a lehetőségeket, amelyeket a BASIC nyújt.

A rutinok 10-es meghívó sorában a következő paramétereket találjuk:

ES és US a nyomtatásra kerülő első és utolsó képernyősor száma;
EO és UO a nyomtatásra kerülő első és utolsó képernyőoszlop száma;
TB a tabulálás mértéke.

A képernyőn tehát behatárolható az a terület, amit ki akarunk nyomtatni, és a nyomtatási kép tetszőlegesen tabulálható.

HARDPRINT 1

(1. program)

A képernyő inverz karaktereit normál karakterekként nyomtatja. Az 5000-es sor változóit a gyorsabb futás érdekében kell bevezetni. Közöttük D a képernyő bal felső sarkához tartozó tárcím. A nyomtató megnyitása után 5035 az aktuális képernyősort és — az F változóban rögzítve — azt a tárcímet adja meg, amelyiknek a tartalmát ebben a sorban elsőként ki kell olvasni.

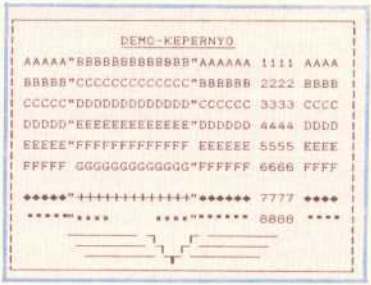
```
10 ES=0:US=24:EO=0:UO=39:TB=5:GOSUB5000
20 WAIT203,63:END
4990 :
4995 REM: HARDPRINT 1
5000 A=32:B=64:C=128:D=1024:UM=0
5030 OPEN4,4,UM
5035 FORY=ESTOUS:A$="":F=D+Y*40
5050 FORX=F+EOTOF+UO:Z=PEEK(X)
5055 IFZ<ATHEN5090
5065 IFZ<BTHEN5095
5070 ONZ/AGOTO5065,5085,5090
5080 Z=Z-C:GOTO5055
5085 Z=A+Z:GOTO5095
5090 Z=B+Z
5095 A$=A$+CHR$(Z)
5100 NEXT:PRINT#4,TAB(TB)A$:NEXT
5105 CLOSE4:RETURN
```

1. program

5050 ciklusa olvassa végig a sort a megadott oszlophatárok között. A Z képernyőkódokat az 5055 . . . 5090 sorok transzformálják ASCII kódokká és helyezik el ugyancsak a Z változóban. Az inverz jelek kódját 5080 a megfelelő normál jel kódjává alakítva visszaküldi a transzformáló sorok elejére.

Egy-egy sor karaktereit 5095 az A$-ben fűzi össze, amit 5100 tabulálva küld a nyomtatóra. Végül 5105 zárja a nyomtatót.

A Z változó kettős felhasználása, valamint az IF . . . és az ON . . . GOTO utasítások kombinációja egyaránt a gyorsabb futást szolgálja. A nyomtatási kép a *3. ábrán* látható.



3. ábra

HARDPRINT 2

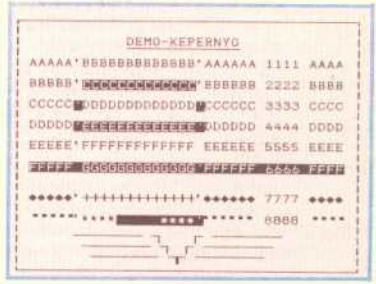
(2. program)

Az inverz karaktereket is nyomtatja, de a kettős idézőjelek helyett egy felső vesszőt ír.

A rutin magját az előző program sorai alkotják, de 5000-ben két újabb változó kapott értéket, 5095 pedig egy feltételes utasítással bővült, ami az inverz jelek feldolgozási útját jelöli ki.

Négy újabb sor egészíti ki a korábbiakat. 5060 és 5075 írja át a kettős idézőjelek képernyőkódjait. Az utóbbiban Q értéke is megváltozik, ha inverz jel feldolgozása következik. 5110 visszaállítja a Q értékét, a jelet pedig egy INVERZ BE (CHR$(18)) vezérlőkaraktert követően fűzi be az A$-be; egyúttal megvizsgálja a következő karakter képernyőkódját. Ha normál karakter következik, akkor egy INVERZ KI (CHR$(146)) vezérlőjelet is fűz az A$-hez. 5095 nem igényel magyarázatot.

Látható, hogy a normál jelek feldolgozási idejét az előző programhoz képest mindössze két újabb — elkerülhetetlen — utasítás lassítja, azok is a legrövidebb végrehajtási időt igénylő változatban.



4. ábra

Az inverzben nyomtatandó jeleket csak a legutolsó után követi egy INVERZ KI vezérlőjel, ezért a nyomtatófej mozgása ebben az üzemmódban is folyamatos marad. A nyomtatási képet a *4. ábra* mutatja.

HARDPRINT 3

(3. program)

Az inverz karaktereket és a kettős idézőjeleket is nyomtatja. A program magja most is változatlan, de a kettős idézőjelek

```
10 ES=0:US=24:EO=0:UO=39:TB=5:GOSUB5000
20 WAIT203,63:END
4990 :
4995 REM: HARDPRINT 2
5000 A=32:B=64:C=128:D=1024:E=34:Q=0:
     UM=0
5030 OPEN4,4,UM
5035 FORY=ESTOUS:A$="":F=D+Y*40
5050 FORX=F+EOTOF+UO:Z=PEEK(X)
5055 IFZ<ATHEN5090
5060 IFZ=ETHENZ=39
5065 IFZ<BTHEN5095
5070 ONZ/AGOTO5065,5085,5090
5075 Q=1:IFZ=162THENZ=167
5080 Z=Z-C:GOTO5055
5085 Z=A+Z:GOTO5095
5090 Z=B+Z
5095 ONQGOTO5110:A$=A$+CHR$(Z)
5100 NEXT:PRINT#4,TAB(TB)A$:NEXT
5105 CLOSE4:RETURN
5110 Q=0:A$=A$+"■"+CHR$(Z):IFPEEK(X+1)<C
     THENA$=A$+"■"
5115 GOTO5100
```

2. program

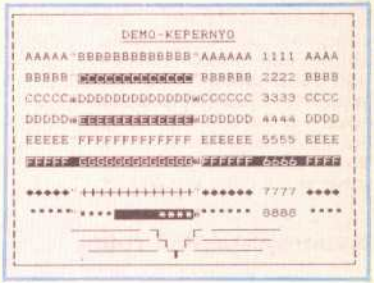
3. program

```
10 ES=0:US=24:EO=0:UO=39:TB=5:GOSUB5000
20 WAIT203,63:END
4990 :
4995 REM: HARDPRINT 3
5000 A=32:B=64:C=128:D=1024:E=34:G=C:
     H=135:Q=1:UM=0
5020 JI$=CHR$(G)+CHR$(H)+CHR$(G)+CHR$(H)
     +CHR$(G)+CHR$(C)
5025 IFQTHENIJ$=JI$:G=255:H=248:Q=0:
     GOTO5020
5030 OPEN4,4,UM
5035 FORY=ESTOUS:A$="":F=D+Y*40
5045 PRINT#4,TAB(TB);
5050 FORX=F+EOTOF+UO:Z=PEEK(X)
5055 IFZ<ATHEN5090
5060 IFZ=ETHENI$=IJ$:GOTO5120
5065 IFZ<BTHEN5095
5070 ONZ/AGOTO5065,5085,5090
5075 IFZ=162THENI$=JI$:GOTO5120
5080 Q=1:Z=Z-C:GOTO5055
5085 Z=A+Z:GOTO5095
5090 Z=B+Z
5095 ONQGOTO5110:A$=A$+CHR$(Z)
5100 NEXT:PRINT#4,A$:NEXT
5105 CLOSE4:RETURN
5110 Q=0:A$=A$+"■"+CHR$(Z):IFPEEK(X+1)<C
     THENA$=A$+"■"
5115 GOTO5100
5120 PRINT#4,A$"■"CHR$(8)I$CHR$(15);:
     A$="":GOTO5100
```

kiírásának három újabb sor és néhány kiegészítés az ára.

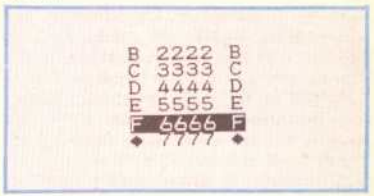
5020 és 5025 állítja elő az első átfutáskor a normál idézőjel IJ$-jét, a második átfutáskor az inverz idézőjel JI$-jét. Közülük az éppen aktuálisat 5060, illetven 5075 továbbítja I$-ként 5120-ba. Itt ki kell nyomtatni az A$ eddigi tartalmát, meg kell szakitani az esetleges inverz üzemet, be kell kapcsolni a „bit image" üzemmódot (CHR$(8)), ki kell nyomtatni az idézőjelet (I$) és visszatérni a karakteres üzemmódba (CHR$(15)), ki kell üriteni az A$-et, és végül visszalépni a ciklus végére. Ha mindezt beépitenénk az A$-be, akkor bizonyos számú idézőjel után STRING TOO LONG hibaüzenettel leállna a program.

Az előzőekből következik, hogy a tabulálási utasitást most az A$ nyomtatásától füg-



5. ábra

6. ábra



getleniteni kell; ezért került az az 5035-ös sorba. Az idézőjelek kivételével minden más karakter feldolgozási módja ugyanaz, mint az előző rutinban, igy az inverz karakterek nyomtatása itt is folyamatos. Az *5. ábra* nyomtatási képén látható az eredmény.

A rutinok bővítése

Mindhárom rutin komfortosságát növelni lehet további négy sor beiktatásával:

5005 IFUS=0THENUS=24
5010 IFUO=0THENUO=39

Igy egyszerűbbé válik a rutin hivása, amikor a teljes képernyőt akarjuk nyomtatni, mert US és UO értékét sem kell megadni.

5015 IFPEEK(53272)=
23THENUM=7

Ezzel a sorral a nyomtató automatikusan alkalmazkodik a számítógép grafikus vagy szöveges üzemmódjához.

```
5Ø4Ø IFSKTHENFORX=
     F+EOTOF+UO:
     IFPEEK(X)=
     ATHENNEXTX,Y:
     GOTO51Ø5
```

A meghívó sorba SK = 1-et írva az üres képernyősorok kihagyására adunk utasítást a nyomtatónak. Hasznos tulajdonság, ha nem a képernyő-elrendezést akarjuk nyomtatásban visszakapni, hanem csak az értékes karakterláncokat rögzíteni, mert így a nyomtató csak annyi sort emel, ahány képernyősor ténylegesen adatokat tartalmaz.

Erre láthatunk példát a *6. ábrán*, ahol a DEMO—KEPERNYO-nek csak egy részletét nyomtattuk ki. A behatárolt területen belüli üres sorokat a nyomtató figyelmen kívül hagyta, így a kiírt sorok tömörebben jelennek meg, mint a korábbi ábrákon.

A fenti sorokkal kibővített rutinok a teljes képernyőt másolják tabulálatlanul, ha a 10-es sor változóinak nem adunk értéket, azaz mindegyik nulla.

Futási idők

Egy képernyőmásolás ideje a nyomtató adottságain kívül nyilvánvalóan nagymértékben függ a képernyőtartalomtól. Tájékoztatásul összefoglaltuk az 1. ábra szerinti képernyőnek és egy inverz jelek nélküli, fele részben betűket, fele részben számokat tartalmazó képernyőnek a másolási idejét másodpercben — a cikkben említett valamennyi rutinnal. A HARDPRINT 3-hoz a leforditott változattal mért időket is megadtuk.

	Demo	Alfanumerikus
Commodore	78	72
PRINTER BASIC	45	14
Simon's BASIC	46	15
HARDPRINT 1	24	22
HARDPRINT 2	32	25
HARDPRINT 3	33	27
HARDPRINT 3. ford.	30	12

Az első számoszlop adatai a BASIC-ben fogalmazott HARDPRINT rutinokra jóval kedvezőbbek, mint a gépi kódúakra. Persze nem a BASIC nyelv táltosodott meg ennyire, hanem ilyen sokat jelent az inverz karakterláncok folyamatos nyomtatása. A reális összehasonlításra a második számoszlop adatai alkalmasak.

(A bemutatott HARDPRINT rutinok és az ezekben alkalmazott megoldásokra épülő képernyőmásoló rutinok a szerző hozzájárulása nélkül nem építhetők be kereskedelmi forgalomba kerülő programokba.)

BARABÁS MIKLÓS

µINFORM

A tartalomleírások az alábbi folyóiratokban megjelent programlistákról készültek:

A folyóirat neve	Kódja
64'er Magazin	64er
Chip Magazin	chip
Commodore Microcomputers	comi
Compute!	cute
Happy Computer	happ
Run /USA/	run
Run /NSZK/	run2
ZX Computing Monthly	ZXCM

A tartalomleíró szóláncokat permutáltuk, a szólánc variánsokat pedig alfabetikusan rendeztük.

A tartalomleírás egy szólánc ból áll, majd a listában ezt követi a forrás megjelölése a folyóirat azonosítójával, a megjelenés dátumával és a cikk előkeréséséhez a kezdő oldalszám és a terjedelem megadásával. A mellékelt lista értelmezéséhez még az alábbiakat kell tudni. A tartalomleírás szóláncában elsőként a téma átfogó megnevezése, utána a számítógéptípus(ok), ezt követően a szűkebben jelölt tartalom meghatározása szerepel, majd esetlegesen néhány, a közleményt minősítő adat (például : cikksorozat).

A forráshely karaktersorozatát nyíl vezeti be, melyet a / jelig a folyóiratok azonosítója, a két / jel között az évszám, folyóiratszám és kötőjellel a kezdő ol-

```
ATARI 400/800 XL
   lemezegysegkatari 1050!programlista
   !(happy-dos ii+/d+ operacios rendsze
   r ->happ/86.03-91/8
ATARI 400/800 XL/XE
   basic programozas!programlista!hibam
   entesites wedge-technikaval
   ->cute/86.05-81/5
ATARI 400/800 XL/XE
   commodore 64!oktatas!programlista!ap
   ple ii!jatekos ora-ido leolvasas
   ->/cute/86.05-3a/6
ATARI 400/800 XL/XE
   commodore 64;128!jatekprogram!progra
   mlista!apple ii!(miami ice)
   ->cute/86.06-34/6
ATARI 400/800 XL/XE
   programlista!keptarolas koalapad/mic
   ropainter formatumban
   ->happ/86.06-84/5
ATARI 400/800 XL/XE
   programlista!programvedelem!load kii
   ktatas ->cute/86.06-96/5
ATARI 400/800 XL/XE
   programlista!vertikalis mozgas!autos
   tart!logikai kapcsolas
   ->hc/86.06-51/2
ATARI 400/800/XL
   programlista!(vbi-spritemover)
   ->hc/86.08-83/2
ATARI 400/800  XL/XE
   programlista!speedscript!felhasznalo
   i karakterek generalasa
   ->cute/86.05-98/3
ATARI 400/800;XL;XE
   basic programozas!programlista!hibaj
   avitas ->cute/86.02-74/3
ATARI 400/800;XL;XE
   commodore 64!jatekprogram!programlis
   ta!apple ii!(high rise)
   ->cute/86.02-49/8
ATARI 400/800;XL;XE
   commodore 16;c64;128;plus/4!jatekprog
   ram!programlista!apple ii!(pasziansz
   ) ->cute/86.01-48/7
ATARI 400/800;XL;XE
   grafikus programozas!programlista!an
   imacio ->cute/86.03-85/6
ATARI 400/800;XL;XE
   jatekprogram!programlista!(switchbo
   x) ->cute/86.03-45/3
ATARI 400/800;XL;XE
   muveletgyorsitas!programlista!adatba
   zis lekerdezes!fuzerkezeles
   ->cute/86.02-58/4
ATARI 400/800;XL;XE
   muveletgyorsitas!programlista!begepe
   lesi segedprogram
   ->cute/86.03-107/3
ATARI 400/800;XL;XE
   programlista!adatvedo rutin!reset/br
   eak kiiktatas ->cute/86.01-109/2
```

```
ATARI 400/800;XL;XE
   programlista!basic programozas!sorto
   rlo rutin ->cute/86.01-112/1
ATARI 400-800;XL;XE
   programlista!speedcalc!alkalmazasi p
   rogramtesztelo ->cute/86.03-e5/13
ATARI 600/800;XL
   jatekprogram!programlista!(dungeons
   of xotha) ->hc/86.01-70/5
ATARI 600/800XL/130XE
   programlista!monitorprogramkent futt
   athato zenei alapstessz
   ->hc/86.05-81/1
ATARI 600/800XL;130 XE
   programlista!operacios rendszer ruti
   nok!hozzaferes basicbol
   ->chip/86.04-170/2
ATARI 800 XL
   programlista!vedes!programlista!list-res
   et modositas automatikus torleshez
   ->happ/86.10-104/1
ATARI 800 XL
   programlista!(mini-dos)!lemezegyseg-
   vezerles!filekezeles
   ->hc/86.03-56/3
ATARI 800 XL
   programlista!cimsor-generalas a 24 a
   lapsor felett ->happ/86.05-83/1
ATARI 800 XL;XE
   programlista!lemezblokk kijeloles
   ->cute/86.03-102/3
ATARI 800XL
   programlista!listazas-lassitas!(slow
   list) ->hc/86.01-94/2
ATARI 800XL/130XE;260/520ST
   commodore 64!sinclair spectrum!terme
   kismerteletes!terminal uzemmod!modem/s
   zoftver kinalat ->happ/86.02-151/3
ATARI ST;800 XL/130 XE
   commodore 64;128!ms-dos!programvedelem!si
   nclair spectrum!q1!apple ii!piaci ki
   nalat(tablazatok)
   ->happ/86.05-117/5
ATARI XL
   programlista!(easy-data)!data-sor be
   gepelesi segedlet ->hc/86.09-45/2
ATARI XL
   programlista!list-stop a rem-sorokba
   n!tomorites ->hc/86.09-4e/2
ATARI XL
   programozasi tippek ->hc/86.10-74/1
ATARI XL/ST
   programlista!120 karakteres futoszov
   eg generalasa ->hc/86.04-70/1
ATARI XL/ST
   programlista!cursor-delete-insert ak
   tivalas az input-hoz
   ->hc/86.04-68/1
ATARI XL/ST
   programlista!digitalis ora generalas
   a a 24 alapsor fole ->hc/86.04-70/2
```

dalszám követi, a végén pedig a közlemény teljes oldalterjedelme áll.

A folyóiratok a SZÁMALK szakkönyvtárában (Budapest,XI.,Szakasits Á. út 68. Nyitva: 8-tól fél 5-ig. Tel.: 853-111/251) is föllelhetők. A kiválasztott anyagról másolat rendelhető az alábbi formában:

SZÁMALK Szakkönyvtára
Budapest, 112. Pf.: 146. 1502
Megrendelem a Mikroszámítógép Magazin 1987/ sz. alapján a következő folyóirat-oldal-másolatokat:
Kód: _____ Példányszám: _____
Kód: _____ Példányszám: _____
Kód: _____ Példányszám: _____

A megrendeléshez csatolom az oldalankénti 8,- Ft-os szolgáltatási díj befizetését igazoló csekkszelvényt.
Dátum, név, pontos cím.

ATARI XL/ST
programlista/informaciosor a 24 alap sor felett (info line)
->hc/8a.04-69/1
ATARI XL/ST
programlista/jatekkeszito segedprogr am ->hc/8a.04-72/8
ATARI XL/ST
programlista/listabegepelesi_segedle t ->hc/8a.05-80/1
ATARI XL/XE
commodore 64/programlista/sakkfeket-s zamitogepes jatek ->happ/8a.11-61/3
ATARI XL/XE
grafika/matematika/programlista/forg astestek/turbo-basic
->happ/8a.09-78/2
ATARI XL/XE
grafika/programlista/dia-show/
->happ/8a.11-65/2
ATARI XL/XE
pascal/compiler kinalat/ertekeles
->happ/8a.09-113/3
ATARI XL/XE
programlista/25o-karakteres/25o-kara kteres futoszoveg generalasa
->happ/8a.10-84/1
ATARI XL/XE
programlista/nano-dos/segitsegu fil ek gyors betoltese
->happ/8a.12-125/3
ATARI XL/XE
programlista/hasznos segedprogramok
->hc/8a.07-43/3
ATARI XL/XE
programlista/hosszu basic programok betoltese gepkiakadas nelkul
->hc/8a.09-49/2
ATARI XL/XE
programlista/katalogusbetoltes
->hc/8a.09-47/2
ATARI XL/XE
programlista/programozasi trukkok/(t urbo-windows) ->happ/8a.09-88/1
ATARI XL/XE
zene/(soundmachine)/ismertetes/29
->happ/8a.12-15a/1
ATARI
basic-valtozatok teljesitmeny-osszeh asonlitasa ->happ/8a.02-96/2
ATARI
cikksorozat/grafikus programozas/ani macio ->happ/8a.01-91/3
ATARI
cikksorozat/grafikus programozasi jat ekprogram keszites/programlista
->happ/8a.02-98/3
ATARI
commodore/compute'/programlista/appl ebeviteli utmutato
->cute/8a.02-114/10

GÉPÉPÍTŐK FIGYELMÉBE

Lapunk 1987/8. számában közzétettük egy Z80 alapú mikroszámítógép leírását, kapcsolási rajzát. Aki meg is akarta építeni, esetleg vásárolni a gépet, az a szerzőtől megrendelhette a kész mikrót vagy a szükséges alkatrészeket. Az érdeklődésre jellemző, hogy 54 db nyomtatott áramköri lap, 14 db építéshez szükséges kit és 16 db teljes kiépítettségű gép talált gazdára.

Úgy véljük, hogy az ilyen szolgáltatásra továbbra is szükség van. Szerkesztőségünk ezért felvette a kapcsolatot a Kandó Kálmán Villamosipari Műszaki Főiskolán működő KASZKAD Kisszövetkezet óbudai PÓLUS szakcsoportjával, amely elvállalta, hogy kívánságra megküldi

— a cikkeinkben konkrétan jelzett esetekben a nyomtatott áramköri lapo(ka)t vagy a működő áramköri példányokat,
— a leírásokban szereplő integrált áramkörök katalógusadatait, valamint
— adathordozón a programokat.

Ez a szolgáltatás nem általános érvényű: minden esetben az adott cikknél közöljük a megrendelhetőség tényét és a konkrét árakat. A csomag- és levélküldés utánvéttel, a postai és kezelési költség felszámításával történik.

Olvasóink igényeiket levélben jelezhetik az aktuális lapszámot követő szám megjelenéséig. **A szerkesztőség**

A SMARTWORK

A számítógép a műszaki élet egyre több területén válik jól használható munkaeszközzé. Kitűnően bizonyítja ezt az alább bemutatott, nyomtatott áramkört tervező program is.

Az elektronikai fejlesztés fontos lépése, amikor a megtervezett áramkört nyomtatott áramköri lapon — NYÁK-on — valósítjuk meg. A NYÁK egy szigetelőlapon vékony rézrétegből (fóliából) kialakított vezetékrajzolat. Erre a lapra szereljük fel az áramköri elemeket úgy, hogy kivezetéseiket a NYÁK-on fúrt lyukakon átdugjuk, és a lyukakat beforrasztjuk. A NYÁK-nak azt az oldalát, ahol az alkatrészeket elhelyezzük, alkatrészoldalnak, ahol forrasztjuk, forrasztási oldalnak nevezzük.

Az alkatrészoldalon célszerű feliratozást készíteni. Ez a dokumentálás során jól használható az áramköri rajz és a NYÁK-on kialakított elrendezés azonosítására. Ha a feliratokat a kész NYÁK alkatrészoldalára rányomtatjuk, akkor vizuálisan is azonosíthatók az egyes áramköri pontok.

A NYÁK-ot alkotó vezető és szigetelő rétegek száma több is lehet (többrétegű NYÁK). Mi a továbbiakban csak a könnyen kezelhető és a gyakorlatban legjobban elterjedt egy- és kétoldalas NYÁK-okkal foglalkozunk.

Egyoldalas NYÁK esetén a vezetékrajzolat a szigetelő hordozólap egyik oldalán helyezkedik el. A kétoldalas NYÁK-nak mindkét oldalán van vezeték, és az egyik oldalról a másikra átmenő huzalokat össze kell kötni. Ezt egyszerűbb esetben a szigetelőlapon fúrt lyukon átdugott és kétoldalról beforrasztott huzaldarabbal, igényes kialakításnál pedig a lyuk átfémezésével, ún. furatgalvanizálással oldják meg.

NYÁK készítése

Az alkatrészeket összekötő, megtervezett vezetékrajzolatot áttetsző fóliára gondosan megrajzolják vagy öntapadós elemekből (különböző vastagságú vonalak, forrszemek stb.) leragasztják. A rajzolat bonyolultságától függően ez vagy 1:1 eredeti, vagy 2:1, 4:1 nagyított méretben történik. Ezt az ún. mesterrajzot lefényképezve készítik el a NYÁK filmjét. A NYÁK gyártása során első lépésként elkészítik a furatokat, majd — amennyiben kétoldalas a NYÁK — a lyukakat átfémezik. Ezután a lemezt fényérzékeny anyaggal vonják be, majd a mesterrajzról készített filmen keresztül megvilágítják. A film csak az átlátszó helyeken engedi át a fényt. Ott a bevonat úgy alakul át, hogy a következő lépésben alkalmazott eltávolító oldószer nem oldja; tehát a rézfólián a vezetékrajzolat egy védőrétegből áll. Most a lemezt rézmaráró fürdőbe helyezik, ahol mindazokról a

```
******** SMARTWORK NYAKTERVEZÖ LEIRASA *********

A funkciógombokkal adható parancsok:

F1: összekötés                F2:  vonaltörlés
F3: potty elhelyezése [F3-t többször lenyomva más-más pöttyöt ad]
F4:  potty  törlése
F5:  vonal  vastagítás  F6:  vonal  vékonyítás
F7:  négyzet  elhelyezése
F8:  összekötés  ismétlése
F9:  blokkmuveletekbe belépés
        és blokk kijelölése
F10: blokkból --> normálba
AltF1: ff/színes váltás
AltF2: háttérfényerővaltás
AltF3: pi/zö - li/ké váltás
AltF4: háttérszínváltás
AltF5: transzp./tömör váltás
AltF6: aktiv oldal színváltás
AltF7: más ablakméret
AltF8: koordináta ki/be
AltF9: pontrács be/ki
<-: vonaldarabka törlés
Ctrl C: vonalhúzás megszakítása

   1 lépéses mozgások
HOME-------------balra-fel
  ▲ --------------fel
PgUp-------------jobbra-fel
  ▶ --------------jobbra
PgDn-------------jobbra-le
  ▼ --------------le
END--------------balra-le
  ◀ --------------balra
   SHIFT-tel 10 lépés

Több-betüs parancsok: -előtte ENTER-rel COMMAND állapotot elérni!

LOAD <név>: nyákterv betöltés SAVE <név>: nyákterv mentés
PAD <méret/forma>: pottyméret megszabás
DD <meghajtó:\útvonal>: a:,c: váltás
DIR: könyvtár képernyőre -esetleg dir <filenév>: filenév szerint keres
COLOR: rétegek szinkombinációjának változtatása
FILL: telefóliázás -FILL: telefóliázás
CLEAR: munkaterület,vagy kijelölt BLOKK törlése
CLEAVE <irány> <raszterszám>: helycsinálás
SIP <irány> <lábszám>: egy sor potty elhelyezése
DIP <irány> <lábszám> <sortávolság>: Két sor potty elhelyezése
     [irányok]:
                    ▲
                    n - north (felfele)

          ◀ w - west (balra ) e - east (jobbra ) ▶

                    s - south (lefele)
                    ▼
-és+: oldalváltás
MOVE: munkaterület,vagy kijelölt BlokK mozgatása
COPY: munkaterület,vagy kijelölt BlokK másolása,ismétlése
VIGYÁZAT! E parancsok kiadásakor a kurzor helye adja az új terület
          bal alsó sarkát!
STAT: az adott nyák jellemzőinek képernyőre iratása(méret,furatszám,stb)
? vagyHELP: segítség a képernyőre
JUMP (X.x,Y.y): ugrás a megadott koordinátákra
TEXT (irány): szöveg,karakter irás
ENTER: parancs indítása      QUIT: vissza a DOS-ba
BACKSPACE: karakter törlés ESC: válasz törlés
Jó munkát!
```

1. ábra. A NYÁK-tervező szerkesztőparancsai

részekről, amelyeket nem fed védőbevonat, a réz lemaródik. Az így létrejött vezetékrajzolatról eltávolítják a maratás elleni védőréteget, majd a rézfelületet oxidáció ellen és a jobb forraszthatóság elérésére vegyi úton önozzák. Ezután már csak a NYÁK méretre vágása és az ültetés van hátra.

A NYÁK-tervező program

A Wintek cég által kifejlesztett SMARTWORK program a mesterrajz és az elrendezési rajz elkészítésében nyújt segítséget. IBM PC kompatibilis számítógépekre készült, használatához színes, grafikus megjelenítő szükséges.

A SMARTWORK három programból áll:

EDIT: a NYÁK megtervezését, szerkesztését végzi. Ezt a programot alább egy kissé részletesebben ismertetjük.

DOT: a megtervezett NYÁK-ot mátrixnyomtatón kinyomtatja. A nyomtatás lehet egyszeres vagy kétszeresre nagyított méretű, és elsősorban dokumentációs célokra használható.

PLOT: a megtervezett NYÁK-ot grafikus plotteren kirajzolja. A rajz mérete megadható. Ez a rajz szolgál a filmkészítés alapjául mint fotózható mesterrajz.

Az EDIT program a képernyőn különböző színekkel jeleníti meg a NYÁK három rétegét: a felirati, az alkatrész- és a forrasztási oldalt. Ezek közül mindig egy, a legfelül lévő az aktív.

A program 254×406 mm méretű munkaterületen dolgozik. A képernyővel mint ablakkal mozgunk ezen a területen, azaz mindig csak egy részét látjuk.

A jobb tájékozódás érdekében a munkaterületen raszter osztású pontrácsozatot jeleníthetünk meg, és koordinátákat is használhatunk. A munkaterületen az irányt a program az égtájak kezdőbetűivel adja meg. Egy aktuális pontot a kurzorral lehet kijelölni.

A program a vezetékrajzolatot négy alapszimbólumból alakítja ki:

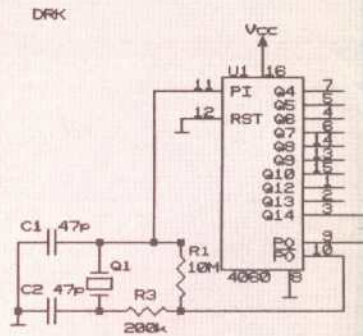
VÉKONY VEZETÉK: a rajzolat két pontját köti össze vékony vonallal; ezek az áramkör (és a NYÁK) jelvezetékei.

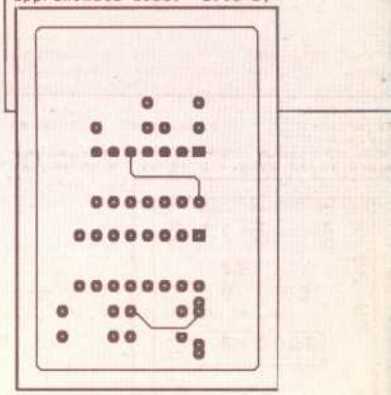
VASTAG VEZETÉK: a rajzolat két pontját köti össze vastag vonallal; ezek az áramkör (és a NYÁK) föld- és tápfeszültség-vezetékei.

Az egyik szomszédos pontból a másikba a vezetékek csak nyolc irányba haladhatnak: egymásra merőlegesek (észak—déli, kelet—nyugati) vagy átlósak lehetnek.

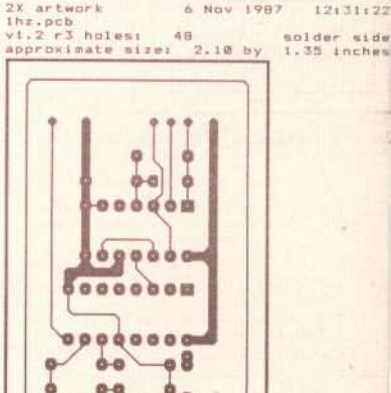
A vastag vonal alapeleme a NÉGYZET, ami a vastag vezető egy négyzet alakú da-

2a. ábra. Kapcsolási rajz

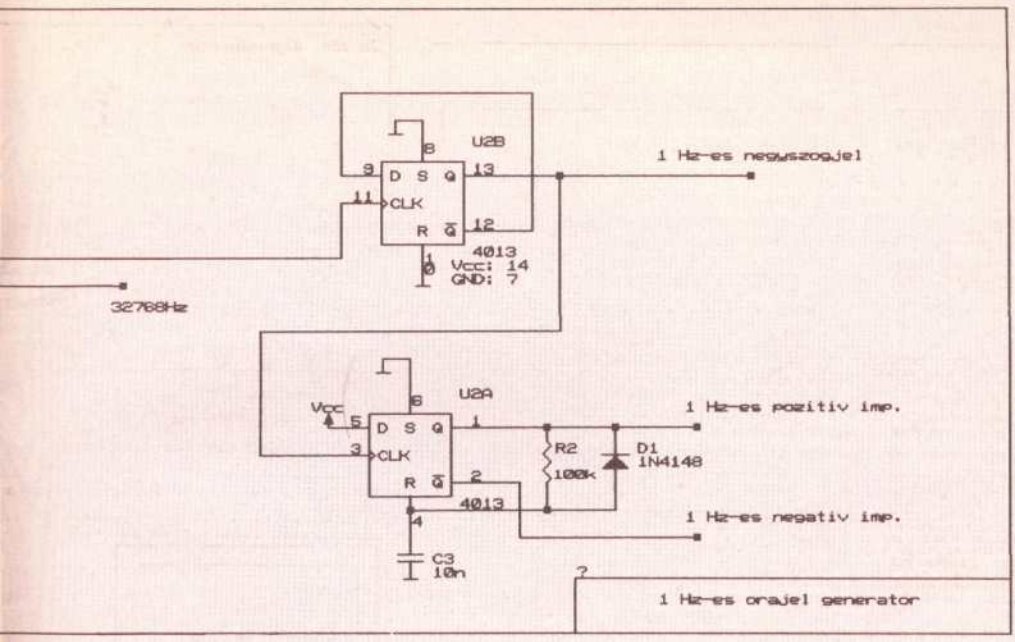


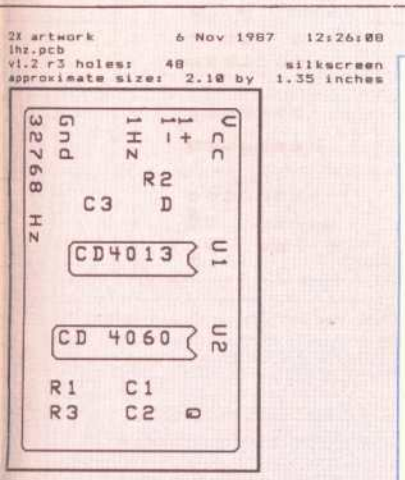


2b. ábra. NYÁK-terv, alkatrészoldal



2c. ábra. NYÁK-terv, forrasztási oldal





2d. ábra. NYÁK-terv, beültetési vázlat

rabja. Segítségével nagyobb vezető rézfólia-felületeket alakíthatunk ki.

A PÖTTY jelöli ki a NYÁK furatait. Ez az elem a NYÁK mindkét oldalán megjelenik.

Kijelölhetjük a NYÁK-terv egyes részeit, az ún. blokkokat. Ezeket mozgathatjuk, ismételhetjük, törölhetjük, ami nagymértékben növeli a szerkesztés gyorsaságát.

A NYÁK tervezése során háromféleképpen adhatunk parancsot;

— a klaviatúra kijelölt billentyűivel,
— a parancssorba írt parancsokkal,
— a kurzorpozicionáló egérrel.

A szerkesztőparancsokkal a NYÁK rajzolatát és a feliratozást készíthetjük el. A parancsok összefoglaló és a szerkesztés közben is használható leírását az *1. ábra* tartalmazza.

A szerkesztés eredményeként kialakult rajzot egy fájlban tárolhatjuk, amely ismét a szerkesztőbe tölthető.

A program használata

A NYÁK-készítés első műveleteként az EDIT programmal interaktív módon megtervezzük a NYÁK-ot. A tervet tartalmazó fájl lesz a tervet mátrixnyomtatóra küldő DOT program bemenete. (A kinyomtatott tervet ellenőrzésre és dokumentációs célokra használhatjuk fel.) Ugyanez a fájl képezi a fotózásra alkalmas rajzolatot plotteren elkészítő PLOT program bemenetét is. A megrajzolt terv fotózásával készül a NYÁK-gyártás alapjául szolgáló mesterfilm.

A SMARTWORK alkalmazásának előnyei:

— a program azonnal szolgáltatja az áramkörhöz tartozó NYÁK dokumentációját,

— az esetleges módosítás a szerkesztőbe való ismételt betöltés után könnyen elvégezhető,

— egyszerűen kialakítható a gyakran használt áramköri részletek NYÁK-terveit tartalmazó modulkönyvtár, amiből nagyon gyorsan állítható össze egy új kapcsolás NYÁK-terve,

— a NYÁK dokumentációja a hagyományos nagyméretű, sok helyet foglaló és kényes mesterfilm helyett egy mágneslemezre zsugorodik,

— sokkal hatékonyabbá és gyorsabbá válik a tervezés.

A program működésének illusztrálására mutatjuk be a *2. ábrán* egy praktikus kapcsolás NYÁK-tervét. Érdekességként megemlítjük, hogy a kapcsolási rajz is számítógépen készült. Az áramkör egy, a digitális órák legtöbbjében megtalálható, 32 768 Hz-es kvarc frekvenciáját osztja le 1 Hz-es négyszögjellé, illetve 1 Hz-es pozitív és negatív impulzussá.

A SMARTWORK NYÁK-tervező csupán egy, és korántsem a legkorszerűbb, leghatékonyabb program ezen a területen. Hogy mégis ezt mutattuk be, annak viszonylag széles körű elterjedtsége és igen egyszerű kezelhetősége az oka.

Bár az előzőekből nyilvánvaló, mégis hangsúlyozni kell, hogy a program használatához nyomtatóra és plotterre van szükség. Mivel a PLOT program bemeneti fájlja a plotter vezérlőparancsait tartalmazza ASCII karakteres alakban, belőle viszonylag egyszerűen előállítható az esetleg rendelkezésre álló NYÁK-on a lyukakat fúró, koordináta-fúrógépet vezérlő lyukszalag.

MEGJEGYZÉS. A cikkben szereplő NYÁK, valamint a kész áramkör a szerkesztőségnek címzett levélben utánvéttel (30,— Ft) megrendelhető. A NYÁK ára 50,— Ft, a kész áramkör ára 250,— Ft.

DR. KÓNYA LÁSZLÓ

Közületek figyelem!
Mikroszámítógépet akarnak vásárolni?
Tájékozódjanak a naprakész piaci helyzetről!
Díjtalan ismertető!
MESZ
Számítástechnika
1368 Budapest, Pf. 193.

Primós harc

Ahogy a mezőgazdaság korszerűsödésével mindinkább múzeumi tárggyá lett elődeink ekéje, ahogy a modern orvostudomány lassanként elfelejti a hajdani felcserek gyógyító, műtő eszközeit, éppúgy a feledés homályába vész lassacskán a számítógépek világában úttörő szerepet betöltő berendezések, gépek sora.

A jövő útja a jelenleginél is sokkalta nagyobb mérvű miniatürizálás úgy, hogy közben az adott gép az óriásoknál ezerszer többet tud, sok tizezerszer bonyolultabb feladatokat old meg.

A honi piacon, az iskolaszámítógépek kategóriájában elsők között szereplő Primo például túl van már végnapjain. Enyhén botrányos körülmények között, szinte elkótyavetyélték utolsó néhány száz darabját a végkiárusításon. Ismerve a hazai számítógépek árait — még akkor is, ha a külföldről behozottak lényegesen drágábbak —, nyugodtan kijelenthetjük: a Primo a megvásárolható gépek kategóriájába tartozott. Sokat nem tudott, de nincs is mindenkinek annyi pénze, hogy tiz-húszezres vagy még ennél is drágább mikroszámítógépet vegyen magának. Az alaptudáshoz az olcsóbb számítógép is elegendő.

Végül is a Primót nem gyártják. Ugyanakkor jó néhány otthonban játszanak, tanulnak rajta, számos iskolában — nem tudván előre gyászos végét — az oktatást könnyitendő megvásárolták, majd szoftver hiányában a tanári szekrénybe, a szertár polcára dugták. Talán nincs az országban egyetlen olyan általános iskola sem, ahol ne lenne számítógép. Ha csak egy is, de az gazdagítja a taneszközállományt. Vajon készült-e felmérés, hogy ezek között hány Primo létezik még, és a meglévőkből hány tölti be eredeti szerepét, vagyis segíti a számítástechnikai ismeretek megszerzését, alkalmazását? Azt hiszem, a kérdésre csak becsült adatokkal szolgálhatnánk, márpedig ennek nem sok értelme lenne. Mindenesetre a Commodore-ok, TV-Computerek, Enterprise-ok világában „nem rúghat labdába" az egyszerű, gyerekcipőben járó — és maradó — Primo. Azzal, hogy nem gyártják, hogy szoftvert nem készítenek hozzá, a meglévők is óhatatlanul halálra ítéltettek.

Vagy mégsem? Néhány lelkes Primo-tulajdonos úszik az ár ellen. Mint a megszállottak, elhatározták, hogy nem adják fel. Nem akarnak hűtlenek lenni egykori kisgépükhöz — mellette nem egynek más, komoly, nagy teljesítményű számítógépe is van —, melyen megtanulták az alapokat. Vállalják akár a számítógépes társak gúnyolódását, csúfolkodását is. Erre a legjobb bizonyíték, hogy egy esztendeje a HCC klub keretében megalakult — csaknem utolsóként — a Primo klub. Állandó székhellyel! A főváros szívében, az Almássy Téri Szabadidőközpontban havonta egy este — a hónap második hétfőjén — találkozhatnak egymással a primósok. Ha egyszerre csak két-három ember jön össze, még az is több, mint a kapcsolatok teljes hiánya. Somogyi György, a Primo klub titkára önmagukról, céljaikról a következőket mondja.

— Az elmúlt évben a Mikroszámítógép Magazinban olvastuk a primósok toborzására szóló felhívást. Öten összejöttünk. E szám is bizonyítja: nem voltunk sokan. Mindenesetre elhatároztuk, hogy együtt maradunk, s aki közénk áll, azt szívesen fogadjuk. Tehát a vállalkozási szellem már megvolt, a hogyan tovább-ot azonban nem tudtuk, meg azt sem, hogy hol tartjuk majd az összejöveteleinket. Azt azonban célul tűztük ki — és ehhez mindmáig tartjuk is magunkat —, hogy a Primo klub ingyenes lesz a tagságnak; ami tőlünk kikerül, azért a primós nem fizet. Fontos döntés volt ez, hiszen akinek Primója van, annak biztos, hogy nincs pénze. És ezt a szegényt még mi is terheljük anyagilag? Nekünk az a feladatunk, hogy segítsük egymást, gazdagítsuk a szerényke, csekélyke programok tárházát, s nem az, hogy ezért még pénzt is elfogadjunk.

Szóval a Magazinban közzétettünk egy programot, amihez azonban a lényegesebb adatok hiányoztak. Kértük, hogy akit érdekel a megoldás, írjon. Erre azért volt szükségünk, hogy címeket kapjunk, és azokat listába vegyük. Nem voltam rest, valamennyi levélírót megkeresem. Felajánlottam, legyen az alakuló klubunk tagja. Így mára mintegy 50 névvel és címmel foglalkozunk, tartunk csaknem rendszeres kapcsolatot.

Milyen szolgáltatásokat nyújtunk? Ha kérdeznek, igyekszünk legjobb tudásunk, felkészültségünk szerint válaszolni. Egyetlen levelet sem hagyunk felelet nélkül. Tanácsokat adunk, ötleteket továbbítunk. A legegyszerűbb programoktól a legbonyolultabbakig, a játékoktól a komoly oktatóprogramokig mindent archiválunk. Ha valaki kéri, szívesen másolunk ebből a kollekcióból. Aki ideadja nekünk a programját, számol azzal, hogy primós társa ingyen és bérmentve, igény szerint lemásolja. Nem szégyellem ezt bevallani. Másként ugyanis nem juthatnánk szoftverhez.

Néhányan annyira átépítették már eredeti gépüket, hogy gyártója sem ismerne rá. Nos, a minket megkeresőknek szívesen adunk felvilágositást a Primo továbbfejlesztéséről. A gyártó úgy hagyott magunkra, hogy nem volt botkormány-csatlakoztatási lehetőség, nyomtató- és lemezegység-csatlakoztató. A klubtagok közül ezt néhányan megoldották, és az érdeklődőknek odaadjuk a leirást. Sőt, a Primo szervizéről is adunk felvilágosítást.

Nyugodtan kijelenthetem, hogy bár kis létszámú klub a miénk, de országos hatású, hiszen a legtávolabbi zugokból is kapunk leveleket. Bárki közénk állhat, tagdíjat nem szedünk, viszont a hozzánk belépőnek — mint HCC klubbélinek — a Neumann János Számítógéptudományi Társaság tagjának kell lennie.

Eddig egyetlen nyilvános szereplésünk volt, a Mikro '87-en, nem számitva az ittenit, az Almássy téren. A Compania szervezet ad otthont nekünk, az ő klubtermében találkozunk egymással. Nem tagadom ugyanakkor, hogy valójában inkább levelező klub vagyunk, s a listánkon szereplő mintegy 180 program nem annyira személyes találkozásokon, mint inkább a posta útján cserélődik.

Szervezett programokra, előadásokra gondoltunk már, de egyelőre nem tartunk ilyeneket; igazságtalanság lenne a levelező partnerekkel szemben, ha őket a távolság miatt kizárnánk ezekből a megmozdulásokból. Marad tehát a szakirodalom-ajánlás, a tanácsadás, a programmásolás.

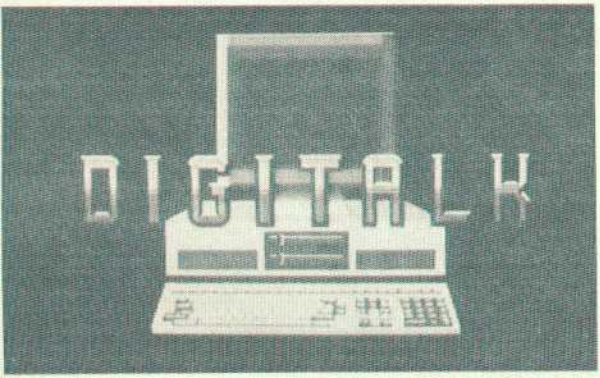
Hogy miért működtetjük a klubot? Mert hisszük, hogy többen többre megyünk! Nem tehetünk arról, hogy a Primo gép gyártását megszűnt. Viszont mi, a tulajdonosok ragaszkodunk ehhez a kezdetleges számítógéphez. Magyar gyártmány, és szeretnénk, ha nem kerülne végleg a süllyesztőbe. Oktatóprogramjainkat használhatják orosz, angol, német, matematika, földrajz, történelem oktatásához. A VII. kerületi Dob utcai általános iskola aktív kapcsolatot tart velünk, ott ki is próbálják, hasznositják munkáinkat. Tehát a kikapcsolódáson kívül segiteni is tudunk.

És még valami. Optimisták vagyuk. Hisszük, hogy egyszer feltámad, újra él majd a Primo. Az ipari termelésben, például egy szerszámgép programozásánál lehetne használni, no meg az irodai munkánál azért is, mert valamennyi kisbetűt irja. Talán rájönnek előnyeire azok, akik ma nem sokra becsülik. Addig meg mi tartjuk a frontot . . .

KRASZNAI ÉVA

Az alapoktól a távoktatásig

A sajtó nagyhatalom — tartja a mondás —, ám azt senki sem vitathatja, hogy az utóbbi két évtizedben óriási konkurenciaként lépett fel a televízió e hatalmi kérdésben. Egy-egy műsor nézettsége — felmérések szerint — messze túlhaladja a lapok példányszámát, s ennek megfelelően az olvasottságát.



A televízió szórakozást, kikapcsolódást, tanulást, okulást nyújt kicsinek és nagynak. Igyekszik a legszélesebb rétegeket, érdeklődési köröket kielégíteni, a legkülönbözőbb igényeknek megfelelni. E cikk nem hivatott a hatékonyságáról vitát indítani, csupán tényként fogadja el, hogy a televízió olyan vállalkozásoknak is teret ad, melyek ezreket, tizezreket mozgósítanak, s ezt a tömegvonzást talán nem is lehetne elérni mással, mint képi-hangi megelevenítéssel.

Az elmúlt néhány esztendőben a számítástechnika is teret kap a televízió műsoraiban. Így lehetőség nyílik arra, hogy olyanokhoz is eljusson ez az „ördöngös tudomány", akik különben kellően ódzkodnak minden ismeretlentől, a hagyományostól eltérőtől.

Hegyi István — számítástechnikai műsorok szerkesztője — elkötelezett és lelkes híve e témának. A közelmúltban zárult például a Digit-alk című 20 részes sorozata. E sokakat érdeklő műsor kapcsán kerestük meg, kérve, számoljon be arról, milyen hatást gyakorolnak a nézőkre ezek az adások.

— Debütálásunk egy máig tartó műsorral kezdődött. A Mi és a számítógép című havi magazinunk első adása 1983-ban volt. A témák változnak, igyekszünk aktuálisak lenni. Ebben a műsorban foglalkoztunk a klubok életével, a szakkörök mindennapjaival, az iskolaszámítógép-mozgalommal. Ugyanakkor a szakmai újdonságoknak is mindig teret adunk. Sőt, több mint egy esztendeje már az adás végén programot sugárzunk Commodore, illetve Sinclair gépekre. Átlagosan 250 ezren tekintik meg egy-egy magazinműsorunkat, s hogy soknak avagy kevésnek nevezhető-e ez a szám, az nézőpont kérdése. Sajnos, változó adásnapon, változó időben kapunk helyet a televízióban, ennek megfelelően nem is számíthatunk stabil nézőközönségre. Ugyanakkor nincs egyetlen olyan klub, tanfolyam, képzés sem, ahol ekkora hallgatóságnak lenne hely, illetve ilyen nagyszámú érdeklődőnek nyújthatnánk információt. És még valami: kezdetben az ifjabb korosztálynak készítettük a műsort, de az eltelt évek alatt kiszélesedett a kör, s az idősebbek is bekapcsolódhattak műsorunkba. Ez is örvendetes. Meg az is, hogy olyanok is bekapcsolják adásunkkor készüléküket, akik már igazi profik, az alapokat automatikusan tudják, viszont rácsodálkoznak egy-egy régi ismeretre, amit már elfelejtettek. Tehát nekik is nyújthatunk információkat.

A TV BASIC volt a következő sikeres akciónk. (Képünkön a „Grafikai lehetőségek" adás szakértője, Kiss Donát.) Ennek célja a BASIC nyelv távoktatása, tulajdonképpen sikerült. Tizenhat adásban — plusz az ehhez megjelent könyvben — rögzítettük a BASIC nyelvről tudható ismereteket. Örömmel tapasztaltuk, hogy 300 ezren (!) néztek bennünket rendszeresen, s 90 ezer példányban fogyott el kiadványunk. Az adással egy időben mikroklubok nyitották meg kapuikat, s tanfolyamunk végén a jelentkezők vizsgát tehettek a BASIC nyelvből. Tizezren jelentkeztek az országból, számuk hatezerre csökkent, míg végül háromezren tettek sikeres vizsgát. Ismét a kérdés: sokan vagy kevesen? Ha azt számítom, hogy egy féléves tanfolyam után háromezren vizsgáztak a számítógépes nyelvek valamelyikéből, akkor azt mondhatom, hogy sokan . . . Hogy miért nem még többen? Amiért Magyarországon nem lehet sikeres a távoktatás. És ez az érdekeltség hiánya! Ugyanis a TV BASIC bizonyítványnyal senki sem helyezkedhetett el a számítástechnika területén. Még akkor sem, ha a vizsga valóban meglehetősen szigorú volt. A '85-ben megismételt sorozat után további ezerkétszáz hallgató szerzett — fel nem használható — papírt.

1985-re jutottunk el oda, hogy a mikroszámítógép lelkesítő, tömegeket megmozgató, újdonságerejű hulláma elült. A kuriózumjelleg megszűnt, ezzel együtt apadt az e témakörrel foglalkozó műsorok nézettsége is.

Ismét tömegeket szerettünk volna megmozgatni, újabb híveket szerezni a számítástechnikának. E gondolat szülte a Digit-alk című műsort, melynek első részét 1987-ben sugároztuk. Már nem annyira a gép technikai ismertetése volt a cél; mint inkább a számítógépek alkalmazásának sokszínűségét kívántuk bemutatni. A Neumann János Számítógéptudományi Társasággal és a SZÁMALK-kal közösen állítottuk össze a sorozatot. Nem panaszkodom, de nem volt könnyű . . . Igyekeztünk népszerű tévés személyiséget megnyerni céljainkhoz, s ez sikerült is Vágó István személyében. Indirekt módon kívántunk oktatni, nem olyan oktató jelleggel, mint azt a TV BASIC műsorban tettük. Azt hiszem, ez a népszerű műsorvezető segítségével sikerült is. Ő volt az, aki ugyan érdeklődött a számítástechnika iránt, ám mégis a kívülálló szerepét játszotta. Legnagyobb meglepetésünkre itt sem léptük át a büvös negyedmilliós nézettséget. Nem tagadom, többre számítottunk. Az okokat kutatva kétféle magyarázatot adhatunk magunknak. Egyrészt a már korábban említett tény, vagyis, hogy a mikroszámítógépeknek ma már — szerencsére — megszűnt az újdonság ereje. Másrészt, mivel az alkalmazásra fordítottuk a hangsúlyt, meglehetősen szakmaspecifikus az érdeklődés. Az orvost nem köti le feltétlenül a számítógép mezőgazdasági alkalmazása, s érthető módon ugyanez fordítva is igaz. Mindenesetre ebből a jövőben okulnunk kell.

Az adások után természetesen sok levelet kaptunk. Ezek köszönő, építő, bíráló hangvételűek voltak, de volt köztük gondolatébresztő is. Sokan hívták fel figyelmünket arra, hogy micsoda útvesztők vannak a számítástechnika világában, s a negatív jelenségek egész sorát vetették papírra. A szoftverkészítés anomáliáitól kezdve, a heterogén géppark hátrányain át, az irányítás visszásságáig mindenre kitértek. Nekünk azonban népszerűsítés és nem leleplezés a célunk. Mégis köszönjük ezeket a megkereső leveleket.

Terveink? Élve a televízió nyújtotta lehetőségekkel, kidolgozni a távoktatás feltételrendszerét. Hisszük, hogy ez a jövő útja.

—i —a

Kis programok még írhatók szét nem bontható, monolitikus egységként, melyen mindössze egy ember dolgozik. Ahogy nő a program, egyre nehezebb lesz az áttekintése, megértése. Jó lenne az egészet kisebb, áttekinthetőbb *részekre bontani* úgy, hogy egy-egy rész felhasználásához ne kelljen annak minden apró részletét áttanulmányozni, megfejteni, csak ami feltétlenül szükséges.

Egy feladatot akkor tudunk értelmesen részekre osztani, ha megértettük a lényegét. Ebben a *jó absztrakció* is segít: kiemelhetjük megoldásunkból a lényeget, csak ezt hozzuk a felhasználó tudomására, s a megoldás részleteivel nem terheljük őt. Így, ha később jobb módszert, tökéletesebb, pontosabb eljárásokat találunk, melyek *nem érintik a felhasználói specifikációt, akkor anélkül, hogy a felhasználót ezzel megzavarnánk, kicserélhetjük a belső* megoldást egy jobbra.

Legyen a feladat mondjuk egy *„keresett"* nevű változóban tárolt elem megkeresése egy *„a"* nevű tömbben, melyben, hozzátesszük, a keresett elem legfeljebb egyszer fordulhat elő. A megoldáshoz elegendő lenne két függvény:

```
FUNCTION  bennevan (t: ARRAY [index] OF..., k: ...):
                    BOOLEAN;
          indexe (t: ARRAY [index] OF ..., k: ...):
                    INTEGER;
```

melyeket így hívhatnánk:

```
megtalálta:= bennevan (a, keresett);
  IF megtalálta THEN z:= indexe (a, keresett);
```

Ha e két függvényt így készen kapnánk, nem is nagyon érdekelne bennünket, hogy a tervező hogyan oldotta meg a problémát. Az *1. ábrán* bemutatott két procedúra például teljesen egyenértékű, hiszen számunkra érdektelen az a különbség, hogy az egyik „alulról felfelé", a másik meg fordított irányban keres.

A programozási nyelvekbe tervezőik mégsem építhetnek be minden elképzelhető függvényt, procedúrát: a nyelv így túl terebélyes és emiatt használhatatlan lenne. Az a cél, hogy a nyelv olyan *mechanizmust* tartalmazzon, hogy a programozó a felhasználás konkrét körülményeihez könnyen illeszthető építőelemeket maga szerkeszthesse meg. Az egyik legelterjedtebb ilyen mechanizmus például a *paraméterezhető* szubrutinok (procedúrák, függvények) alkalmazásáé. Paraméterezéssel nő a szubrutin felhasználhatóságának köre. Így például egy előre megírt, tömböket *rendező* procedúra nemcsak egy — mondjuk „t" nevű — tömböt képes fogadni, hanem más nevűeket is. (Lásd a *2. ábrát.*)

Ha a procedúrát olyan *kommentárral* látjuk el, mely lehetővé teszi a használatát mások számára anélkül is, hogy ehhez meg kellene nézni a „belsejét", akkor — akár öntudatlanul is — olyasmit csináltunk, amit specifikációs absztrakciónak nevezhetünk. A *jó kommentár* három részből (állításból) áll:

— az első a procedúrába való *belépés* előfeltételeire,
— a második annak eredményére,
— a harmadik arra vonatkozik, hogy mit *módosít* a procedúra.

Nézzük például a *3. ábrát.* Ha *csak* a felhasználásban vagyunk érdekeltek — és nem megtervezni kívánjuk a „négyzetgyök" nevű procedúrát —, akár el is feledkezhetünk a BEGIN ... END közötti részről.

Az absztrakció és a specifikáció érzékeltetett módszerével nemcsak procedúrák esetében élhetünk, hanem adatoknál is. Tudjuk: az adatok fontos tartozékai azok a *műveletek,* melyek érvényesek (alkalmazhatók) rájuk. Ezekkel lehet létrehozni, módosítani, törölni stb. őket, ezekkel lehet információt szerezni róluk.

Legyen a feladat például egy olyan egész számokból álló, **egész-sor** nevű adattípus létrehozása, melyre legyenek alkalmazhatók az alábbi műveletek:

— új-sor;
— üres-e;
— told;
— első.

Az **új-sor** nevű művelet eredménye egy új, **egész-sor** típusú adat, mely egyelőre üres: még nincs egyetlen egy eleme sem. Ezt így deklaráljuk:

```
FUNCTION új-sor: egész-sor;
```

Azt, hogy egy ilyen **sor** nevű, **egész-sor** típusú adat üres-e vagy sem, ilyen függvénnyel állapíthatjuk meg:

```
FUNCTION üres-e (q: egész-sor): BOOLEAN;
```

Ha a **sor** üres (nem tartalmaz egyetlen elemet sem), akkor
üres-e(sor)=TRUE,
ha a **sor** nem üres, akkor
üres-e(sor)=FALSE.

A **told** nevű procedúra az **egész-sor** típusú adat végét egy újabb elemmel egészíti ki. Ezt a procedúrát így deklarálhatjuk:

1. ábra. Két „egyenértékű" kereső procedúra. A felhasználót gyakran nem érdekli a konkrét megvalósítás

```
PROCEDURE keresl;
BEGIN
 megtalálta:=FALSE;
 i:=1;
 WHILE(i < max+ 1) DO
 BEGIN
 IF(a [i]=keresett) THEN
 BEGIN
 z: =i;
 megtalálta:=TRUE
 END
 i:= i + 1
 END
END;
```

```
PROCEDURE kereS2;
BEGIN
 megtalálta:=FALSE;
 i:=max;
 WHILE(i>0) DO
 BEGIN
 IF(a [i]= keresett) THEN
 BEGIN
 z:=i;
 megtalálta:=TRUE
 END
 i:= i - 1
 END
END;
```

2. ábra. Példa a paraméterezhető procedúrára. Az „akármilyen" nevű programon belül deklaráltunk egy „rendez" nevű procedúrát, melyet a programon belül először a „rendelések", másodszor a „foglalások" nevű aktuális paraméterrel hívunk

```
PROGRAM akármilyen (input, output);
    .
    .
VAR rendelések, foglalások: ARRAY [index] OF INTEGER;
    .
    .
    .
PROCEDURE rendez ( a: ARRAY [index] OF INTEGER);
  BEGIN  ...  END;
BEGIN
  .
  .
rendez (rendelések);
  .
  .
  .
rendez (foglalások);
  .
  .
  .
END.
```

megint egy jó könyvet (B. Liskov and J. Guttag: „Abstraction and Specification in Program Development", The MIT Press, 1986.), melyre néhány érdekes gondolat kiemelésével szeretnénk felhívni a figyelmet.

```
PROCEDURE told (q: egész-sor, e: INTEGER);
        :
```

Az **első** nevű függvény (olyan procedúra, mely értéket ad vissza)
- *megköveteli*, hogy alkalmazásakor a sor *ne* legyen üres;
- az a *hatása*, hogy a függvény felveszi a sor első tagjának értékét és
- *módosítja* a sort.

Ezt így deklarálhatjuk:

```
FUNCTION  első (sor:egész-sor): INTEGER;
        :
```

Figyeljük meg, hogy bár mind a **told**, mind az **első** művelet *módosítja* a **sor** nevű formális paramétert, ezeket *nem* hivatkozási (változó, VAR) paraméterként deklaráltuk — eltérően attól, ahogy ez a *nem* „absztrakt" műveleteknél megszokott. Ennek az az oka, hogy az **egész-sor** nevű — és minden egyéb „absztrakt" — adattípust Pascalban *pointerek* segítségével valósíthatjuk meg: ha ilyen „absztrakt" adatot adunk át egy függvénynek/procedúrának, akkor tulajdonképpen így mindig csak *hivatkozást* adunk. Amikor az „absztrakt" adatot módosítjuk, akkor a *hivatkozott* adatot módosítjuk, nem pedig a paraméterként adott hivatkozást. Így, ha Pascalban valósítunk meg „absztrakt" adatot, akkor az ezt manipuláló procedúrák, függvények számára a paraméter *sohasem lesz VAR paraméter.*

Lássuk most a megvalósítás néhány jellemző részletét:

```
        :
TYPE
   egész-sor = ↑egészsorp;
   egészsorp = ↑egészsorelem;
   egészsorelem = RECORD
                    érték: INTEGER;
                    következő: egészsorp
                  END;
VAR  sor: egész-sor;
        :
```

A fenti deklarációval egy **sor** nevű, nem inicializált változónak helyet biztosítottunk a **stack**ben — a hely éppen egy pointernyi. Az **új-sor** függvényben és a **told** procedúrában gondoskodnunk kell majd arról, hogy ha a **sor** változót létrehozzuk, illetve feltöltjük, akkor a NEW primitívvel (ez a Pascal nyelv egyik *beépített* eljárása) helyet foglaljunk a tárban az új elem számára, illetve ha az **első** nevű függvénnyel „fogyasztjuk" **sor**-t, akkor a Pascal DISPOSE nevű másik primitívjével felszabadítsuk a megfelelő tárhelyet.

Emlékeztetjük még a Pascalban jártas olvasókat arra, hogy ennek a nyelvnek van egy jól ismert, „beépített", „absztrakt" adattípusa: a FILE (a fájl). Megvalósításának részletei — szempontunkból ez a lényeg — rejtve maradnak a nyelv használója előtt. A FILE-hoz a programozó csak a nyelv *beépített* eljárásain (get, put, read, write, eof stb.) keresztül férhet hozzá.

3. ábra. Ha a procedúrát megfelelő kommentárral látjuk el, akkor a felhasználó nem kell, hogy feltétlenül ismerje a BEGIN... END közötti konkrét megvalósítást (az implementációt)

```
PROCEDURE négyzetgyök (x: REAL, y: REAL);
{előfeltétel a belépéshez: x>0}
{a procedúra eredménye: y-ban x négyzetgyöke}
{a procedúra y-t módosítja}
BEGIN
{Ide jön a megvalósítás, az implementáció}
END;
```

Az ilyen „absztrakt" adattípusok különösen hasznosak, mivel a tervező számára lehetővé teszik az adatstrukturálásra vonatkozó — gyakran igen nehéz és felelősségteljes — döntés *későbbre* halasztását: akkorra, amikor a tervező (a megoldásban kellően előrehaladva) már az adatok felhasználásának módjait is kellően áttekintette. (Az implementálás ezzel a technikával elhalasztható a specifikáláshoz, a deklaráláshoz képest.)

Egyes (újabb) nyelvek explicit módon támogatják a felhasználót abban, hogy olyan konstrukciókat hozhasson létre, amelyek *elrejtik* az algoritmusoknak és az adatstruktúráknak a felhasználás szempontjából felesleges részleteit. Ilyen nyelv például a Modula—2 vagy az Ada.

A Pascal eredetileg azzal a feltételezéssel készült, hogy főleg diákok elsősorban *rövid* programokat fognak írni benne. Ezért a Pascalban a programok (eredetileg) *egy egészet*, egy *fordítási egységet* alkotnak, ahol például a procedúrák az ún. globális változókon keresztül kommunikálhatnak egymással. Mivel a nagyobb programok fejlesztéséhez már nagy szükség van a *modulokra* (például az önálló fordítási egységekre) bontásra, terjedni kezdenek a nyelv bővített, továbbfejlesztett változatai (a közös név — a Pascal — megmaradt, de a kompatibilitás már nem teljesen). Érdemes-lesz a Pascal sorsának tükrében figyelnünk, hogyan alakul a 80-as években startolt Modula—2 karrierje.

A MODULA—2 főleg abban nyújt többet a Pascalnál, hogy bevezeti a „MODUL" nevű konstrukciót, mely a részletek eltakarására szolgál. A MODULA—2 például megkülönböztet ún. „definíciós modult" (DEFINITION MODULE) és „implementációs modult" (IMPLEMENTATION MODULE).

A Pascal nyelvet — akárcsak később a MODULA—2-őt — N. Wirth vezette be a hetvenes évek elején. Célja elsősorban az volt, hogy az oktatás számára tervezzen tiszta, világos, elegáns nyelvet. Emiatt természetesen sok olyan dolgot elhagyott, mely a gyakorlati célú, elterjedt nyelvekben szokásos volt (például a kifinomult adatbeviteli és -kiviteli formátumozást, bonyolultatbb fájlkezelést stb.). A Pascal sokak meglepetése „kinőtt az iskolapadból" és üzleti siker is lett. A diákok, akik ma többnyire ezen nőnek fel, vonakodnak áttérni a nehézkesebb (bár kétségtelenül többet tudó) nyelvek használatára.

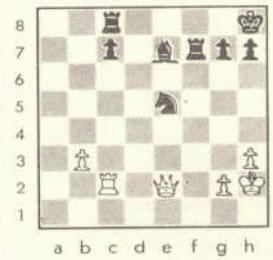
— KÉ —

4. ábra. Az „új-sor" nevű függvény és a „told" nevű procedúra. Figyeljük meg bennük a NEW alkalmazását

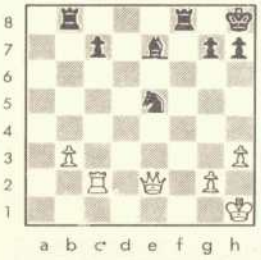
```
FUNCTION  új-sor  : egész-sor;
     VAR  q : egész-sor:
     BEGIN  NEW(q); q↑ :=NIL;  új-sor:=q END;
        {Figyeljünk fel a NEW-ra!}
        .
        .
        .

PROCEDURE told (q:egész-sor; e: INTEGER);
   VAR utolsótag, tag: egészsorp; {egész-sor-pointer}
   BEGIN
     NEW (tag);  {Ezt figyeljük meg!}
     tag↑. érték:=e; {a paraméter értékét átadjuk}
     tag↑. következő:=NIL; {most ez lett az utólagos tag}
     IF üres-e(q) {hívtuk az "üres-e" függvényt}
        THEN q↑:= tag
        ELSE BEGIN
             utolsótag:=q↑;
             WHILE utolsótag↑. következő<>NIL  DO
                   utolsótag:= utolsótag . következő;
             utolsótag↑. következő:= tag
             END;
   END;  {a "told" procedúra vége}
```

A játékfa és kiértékelése 6.

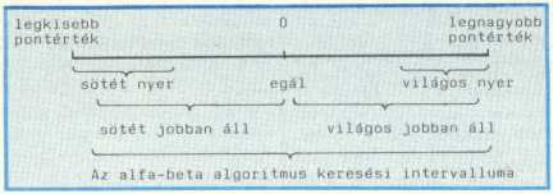
A sakkjátékban ritkán fordul elő olyan eset, hogy a lépésre következő fél azonnal le tud nyerni egy gyalogot vagy még ennél is nagyobb anyagi értékű figurát. Hasonlóan ritka az is, hogy az egyik játékost több mint egy gyalog feladására kényszeríti az ellenfél. Az alfa-béta algoritmus felgyorsításához előnyös lehet a kezdetben minden olyan változat figyelmen kívül hagyása, amelynek pontszáma ezen az intervallumon innen vagy túl esik.



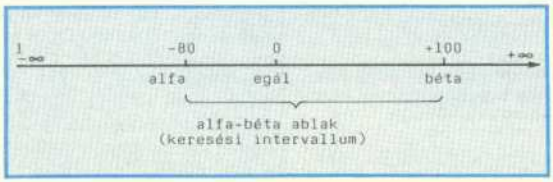
1. ábra. 1.Vxe5? Bf1+ 2.Kh2 Fd6!



2. ábra. 1.Vxe5! Fd6? 2.Vxd6 cxd6 3.Bxc8+



3. ábra



4. ábra

Az alfa-béta algoritmus 100 százalékos biztonsággal talál egy $-\infty$-nél nagyobb pontértékű lépést, de annak is legalább 90 százalékos az esélye, hogy talál olyan lépést, amely nem veszít többet, mint egy gyalogot az eredeti kiinduló álláshoz képest. Így a programnak célkitűzése lehet egy ilyen lépés megtalálása. Az algoritmus megpróbálja az ellenfél olyan lépéseit kizárni, amelyek lehetővé teszik, hogy a program két gyalog előnynél többre tegyen szert. Ezt a két gyalog előnyt mindig az aktuális állástól kell számítani, amit a program éppen elemez. Elképzelhető ugyanis, hogy a programnak van már egy tiszt előnye, amit korábban szerzett. Ezért tiszt előny esetén is tovább számítja a változatokat, és csak az újabb, legalább két gyalogot nyerő lépéseket nem veszi figyelembe.

Az alfa-béta ablak jelentősen gyorsítja a programot, ezért mélyebb kutatásokra ad lehetőséget. Igaz, egyszer-egyszer előfordul, hogy az állásértéket valóságos pontszáma az ablakon kívül helyezkedik el, és a program nem találja meg a számára legkedvezőbb folytatást. Attól függően, hogy a valódi pontérték az ablak alsó korlátja alatt vagy a felső korlátja fölött van-e, előfordulhat, hogy egy rossz lépést jónak, egy jó lépést rossznak értékel. Az *1. ábrán* arra látunk példát, hogy a program egy erősnek tűnő saját lépést vizsgál, és ezt meglépné, nem számítva tovább, mert pontszáma nagyobb az ablak felső határánál. Azért nem keresi az ellenfél cáfoló lépését, mert esetünkben a program a nyereséget két gyalogra limitálja. A *2. ábrán* viszont a világossal játszó program nem találja meg a nyerő lépést, mert az ellenfél válasza túl jónak mutatkozik, és ezért az alfa-béta ablak alsó határa alá esik a pontszám. Mindkét esetben eredménytelennek mondható a kutatás, s az alfa-béta eljárást vezérlő algoritmusnak az ablakot szélesítenie kell. Ezt megtehetjük úgy, hogy egy-két gyalogérték ráhagyásával tágítjuk az ablakot, vagy használhatjuk az eredeti alfa-béta eljárást, teljes szélességben.

Igaz, ilyenkor feleslegesen elemeztük ki az állást alfa-béta ablakkal, és ezzel eredménytelenül ment el az időnk, de a gyakorlat azt bizonyítja, hogy az olyan állások mennyisége, amelyeket az előbbiek miatt

1. lista

```
alfabeta (p:pozicio; alfa, beta: integer);
var
     m, i, t, w: integer;
     lepes : array[1..MAX_SZELESSEG] of integer;

begin
     if (terminalis(p))
        return (visszavesz(p));

    w := lepesgenerator (p);
    m := alfa;
    for i := 1 to w do
        begin
            t := -alfabeta(p,lepes[i] ,-beta,-m);
            if ( t > m )
               m := t;
            if ( m >= beta )
               return (m);
        end;
        return (m);
end.
```

2. lista

```
AlfaBeta(p: pozicio; alfa,beta,melyseg: integer): integer
var
    ertek,sz,t,m : integer;
    lepesek : array[1..MAX_SZELESSEG] of integer;

begin
    if melyseg = 0 then  { terminalis csomopont? }
        return( Pontertek(p) );
    sz := Lepesgenerator( lepesek );
    if sz = 0 then       { nincs legalis lepes? }
        return( Pontertek(p) );

    ertek := - ;
    for m := 1 to sz do
        begin
            t := -AlfaBeta( p,lepes[m], -beta,-alfa,melyseg-1 );

            if t > ertek then
               ertek := t;

            if ertek >= beta then
               return( ertek );

            alfa := Max( alfa,ertek );
        end;

end.
```

újra kell vizsgálni, elenyésző a többihez képest, ahol viszont jelentős megtakarítást érhetünk el.

A *3.* és *4. ábra* jól szemlélteti az alfa-béta eljárás és az alfa-béta ablakkal dolgozó algoritmus értékelési intervallumának különbségét. Esetünkben az alfa-béta ablakkal működő algoritmusban az alfa kezdőértéke −80, a bétáé pedig +100.

Az első sakkprogramokban használt alfa-béta eljárás nem volt alkalmas az ablaktechnika használatára. Ha ugyanis az alfa és a béta nem a −végtelen, +végtelen értékkel indult, akkor előfordulhatott, hogy egy szűk ablak esetén levágta az összes lépést, és nem adott felvilágosítást arról, hogy van-e egyáltalán megfelelő lépés, és annak hozzávetőlegesen menynyi a pontértéke. Ezt a kezdeti programot láthatjuk az *1. listán.* A programokat a későbbiek folyamán úgy alkották meg, hogy ablaktechnika használata esetén legalább annyi információt tudjanak adni: az eddig talált legjobb lépés pontértéke nagyobb-e mint béta, kisebb-e mint alfa, vagy benne van-e az ablakban. (*2. lista*). Nézzük meg konkrétabban, mit is jelent ez a három eset.

1. A pont ≥ béta: a lépés, amelyet a program megtalált, jobb a várt értéknél, de más lépések még jobbak is lehetnek.

2. A pont ≤ alfa: a legjobb lépés rosszabb a vártnál, de más lépések még rosszabbak is lehetnek.

3. alfa < pont < béta: a program megtalálta a legjobb lépést.

Az első esetben, ha az állás pontértéke nemcsak a bétánál nagyobb, de a mattadás pontszámát is meghaladja, akkor nem kell ismét vizsgálni az állást, mert a mattadás a végső cél. A második esetben viszont, ha a pontszám nemcsak az alfánál kisebb, de eléri a vesztes változat pontszámát, amikor a gép kényszerítő módon kap mattot, ezt úgysem tudja kivédeni, tehát hiábavaló nagyobb ablakkal az ismételt keresés.

KOVÁCS P. ATTILA

Azt hiszem, sohasem jutunk a nyári levelek végére. Ezúttal közöljük a második válogatásunkat, így az őszi olvasói levelek a következő számokban kerülnek sorra.

ifj. Pálinkás László, Monorierdő

1987 januárjától a Magazinban sokkal több Spectrum-programot találtam, mint Commodore Plus/4 programot. Én egyetértek ifj. Malak Miklós véleményével: ha a spectrumosok több programot szeretnének, akkor küldjenek ők be. Szeretném, ha ezután minél több Commodore-program jelenne meg a Magazinban, és ha a Magyarországon megjelent játékprogramokról árjegyzéket küldenének.

Hát akkor várjuk a spectrumosok küldeményeit. Az árjegyzékkel kapcsolatban a számítógépboltokhoz kell fordulni, a Novotrade-hez, a Skálához, a Computer—M-hez és másokhoz. Mi árjegyzékek, utánvéttel játékprogramok, alkatrészek és más áruk küldésére nem vállalkozunk. Ezt más olvasóinknak is üzenem.

Gömöri József, Budapest

Köszönöm, hogy közölték cserepartner-kereső hirdetésemet a szeptemberi számban. Segítségével sikerült olyan partnereket találni, akiknek szintén C128-as gépük van, és tudtunk programokat cserélni, egymás programkönyvtárát kiegészíteni.

Együtt örülünk! Mások partnerkereső felhívásait is várjuk.

Csáki Roland, Göd-felső

Rendszeresen vásárolom lapjukat, mert sok érdekes és fontos dolgot találok benne C Plus/4-es gépemhez. De szeretném — és ezzel mások is így vannak —, ha nemcsak kész programokat, hanem részprogramokat is közölnének. Egyébként szívesen kapcsolatba lépnék tapasztalat- és programcsere céljából C16- és C Plus/4-tulajdonosokkal.

Szoktunk részprogramokat is közölni, igaz, nem túl sokat. Javaslom, hogy keresse meg az NJSZT HCC Commodore szekcióját. Találkozóikat rendszeresen a TIT Stúdióban tartják (XI., Bocskai út 37.), minden hónap második csütörtökén, 18 órától.

ifj. Mátyus István, Debrecen

A Magazin 1986. márciusi számában olvastam, hogy a HCC elkészítette C16-os gépre a teljes magyar betűkészlet megvalósításához szükséges hardvert és szoftvert. Én vállalkoznék a NYÁK elkészítésére, csak a szoftvert kellene a HCC-nek megírnia. Kérem, közölje, hogy ez mennyibe kerülne és a C Plus/4 memóriakiosztásában az áramkör milyen változást eredményezne. Azt is szeretném tudni, hogyan lehetnék a HCC Commodore szekciójának a tagja, és mennyi a tagsági díj?

Dr. Simonyi Endre válasza: „Lapunk 1986/2. számának 38. oldalán az áll, hogy a HCC vállalja . . . a . . . megoldás kidolgozását *(a kiemelés most tőlem), nem pedig az, hogy* elkészítette. *Mivel pedig sem a gyártók, sem a forgalmazók, sem az oktatás irányítói azóta sem fordultak hozzánk, mi ezt nem végeztük el. Egyetlen személy részére ezt kidolgozni túlságosan drága lenne. A HCC Commodore Club C64 és C128 típusú gépekkel foglalkozik."*

Azt javasolom, menjen el a HCC Commodore szekció foglalkozására, és ott a klub vezetői részletes felvilágosítást fognak adni a klub működéséről. (Lásd Csáki Rolandnak írt válaszomat.)

Kádár Sándor, 41 201 Litoměřice

Rendszeres olvasója vagyok a lapnak, amit fogalmazhatnék úgy is, hogy a *mi* lapunknak. Sok érdekes cikket, programot, ötletet találtam már a hasábjain. Jelenleg egy C16 számítógép boldog tulajdonosa vagyok. Tudom, hogy ön nem szimpatizál ezzel a géppel, de remélem, segíteni fog, ha tud. Több olyan programom van, amit szeretnék közzétenni, hogy mások is megismerjék. Ez egyszerű is volna, ha lenne nyomtatóm. De sajnos ennek megvételére eddig nem volt lehetőségem. Külföldön élek, itt is dolgozom, ezért egy kicsit nehéz a kapcsolattartás az otthoniakkal. Ebből is fakad, hogy nem jutottam még nyomtatóhoz. Az az elképzelésem, hogy akad olyan C16 számítógép-tulajdonos, akinek van nyomtatója. Én elküldeném neki kazettán a programokat, ő pedig kinyomtatná. Aki vállalja ezt, azt az előnyt élvezi, hogy kézhez kapja és lemásolhatja a programokat anélkül, hogy fáradságos munkával be kellene gépelnie azokat. Ha megfelelő kapcsolat alakulna ki, egyéb programokat is szívesen küldenék: BASIC DATA-író, MOVE rutin, APPEND rutin, turbó kazettához, tárolós oszcilloszkóp stb. Szeretnék egy-két dolgot, gondot szakemberrel megbeszélni. Ezért arra kérném, hogy ha van olyan személy, aki elég jól ismeri a C16-ot, és szívesen segítene nekem, annak a címét írja meg nekem, hogy felvehessem vele a kapcsolatot.

Kérését szívesen közzéteszem. Azt hiszem, hogy levelének megjelenése után néhányan biztosan jelentkeznek, akik készséggel együttműködnek hardver- vagy szoftverproblémáinak megoldásában.

Pichler László, Budapest

Mindig nagy érdeklődéssel várom és olvasom lapjukat, bár nekem nincs személyi számítógépem. A fiam Spectrumján szoktam szórakozni. Mostanában kacérkodom a gondolattal, hogy vásárolok egy 128-as Enterprise-t, de hiába várom, hogy megjelenjen lapjukban valami ismertetés vagy teszt erről az új, még kevéssé ismert gépről. Mellesleg úgy veszem észre, hogy lapjuk — a többi testvérlappal együtt — túlságosan Commodore-centrikus.

Az Enterprise nálunk még nagyon új gép; viszonylag kevés és főleg kevés jó, érdekes és eredeti programot készítettek a gépek új tulajdonosai. Nem tehetünk arról, hogy nagyon sok a Commodore gép a háztartásokban, nem is beszélve az iskolákról. Ezt a helyzetet tükrözi a Magazin programkínálata is. Sorry!

Gergely Zsigmond, Leninváros

Pécsen láttam meg először az Enterprise számítógépet. Jó tulajdonságai miatt rögtön beleszerettem, és mivel nagyon olcsó, remélem, meg is fogjuk venni. De Magyarországon ez elég új gép, és még sehol se láttam róla könyvet vagy bármilyen cikket. Ezért kérem, ha tudnak róla valami érdekeset, vagy programot rá, vagy hogy hol vehetek programokat rá, legyenek szívesek megírni. Annak is örülnék, ha egy Enterprise-szal foglalkozó könyv címét írnák meg, és azt, hogy hogyan juthatok hozzá.

Nem ismétlem meg Enterprise-ügyben az előző levelekre adott válaszomat. Úgy tudom, hogy a géphez nincs kész program, például olyan választék, mint amekkorát a C64-hez kínálnak. Külföldi barátaim azt mondják, hogy ez igen kiváló gép, de a nagyokkal folytatott versenyben megbukott. A gyártó majdhogynem a csupasz hardvert árusította ki igen alacsony áron, és viszonylag alacsony darabszámban, hiszen a gyártás akkor maradt abba, amikor még el sem kezdődött. Ha a pletyka igaz (mondanák Pesten), akkor ehhez a géphez a vásárlók gyári — professzionális — szoftvert hiába várnak.

KOVÁCS GYŐZŐ

IBM PC/XT, PC/AT felhasználói és technikai információs kártya
Szerk.: Éltető László
(Budapest, 1987.
LSI ATSZ,
96 oldal. Ára: 126,— Ft.)

A kötet hasznos információkat tartalmaz a PC DOS, a DOS parancsok, a billentyűzet, a gyári programok használatáról. Külön fejezetek tartalmazzák a BASIC programozási segédletet és a TURBO PASCAL programozással kapcsolatos ismereteket, az IBM PC/XT, AT kódtábláit, a dBASE III és a dBASE III PLUS programozási segédletét.

Weber, M.:
IBM PC 3D-grafika.
Elmélet és gyakorlat
(Budapest, 1987.
IWT — Novotrade,
178 oldal. Ára: 380,— Ft.)

Az olvasó válogatott példák tanulmányozásán keresztül ismerkedhet a grafikus ábrázolás, elsősorban az ún. 3D-grafika (a háromdimenziós ábrázolás) elméletével és gyakorlatával. Ehhez a számtani alapműveletek, illetve a Microsoft BASIC programnyelv ismeretén kívül semmiféle felkészültség nem szükséges.

A kötet első része a grafikus ábrázolás elméletével foglalkozik. Egyszerű, szemléletes példákon keresztül ismerteti meg az olvasót a vektorszámítás, majd a mátrixszámítás alapvető tudnivalóival.

A második rész témája a gyakorlati programozás. Először egyszerű grafikus ábrák készítésére alkalmas példaprogramokat mutat be, majd összetettebb programokkal szemlélteti a számítógépes grafika alkalmazásának lehetőségeit. Részletesen tárgyalja az egyszerű geometriai alakzatok programozását, a szabadon választható irányú írásformákat, az üzleti grafikát, a számítógéppel segített tervezést (CAD) és számos más gyakorlati problémát. A programok az IBM PC és a vele kompatibilis gépeken igen elterjedt Microsoft BASIC 2.0 programnyelven készültek.

Simons, G. L.:
Szakértői rendszerek és mikrók
(Budapest, 1987.
Műszaki Könyvkiadó,
186 oldal.
Ára: 50,— Ft.)

A világ számítógépgyártásában sok összetartó, egymást fedő irányzat figyelhető meg. Ezek között kiemelkedő jelentőségű a mikroszámítógépek egyre növekvő kapacitása és a mesterséges intelligencia, mint sokoldalú piaci realitás.

A mesterséges intelligencia térhódítását az új progracsomagok, nyelvek, fejlesztési eszközök és célorientált gépek számának állandó emelkedése jelzi. A mesterséges intelligencia gyakorlati megvalósítása egyes területeken jelentős számítástechnikai kapacitást igényel. A mikroszámítógépek egyre inkább megfelelnek ennek a követelménynek, sőt, a mesterséges intelligencia mind több modellje már eleve mikrogépen jön létre.

A könyv ismerteti e két fejlődési irányzatot és szemmel látható közeledésük részleteit. Általános képet nyújt a mikroszámítógépek, a szakértői rendszerek és a mikrogépes mesterséges intelligenciák fejlődésének főbb tendenciáiról.

Kepes János:
Mikroszámítógépes grafika.
Grafikai algoritmusok
(Budapest, 1987.
Műszaki Könyvkiadó,
156 oldal. Ára: 55,— Ft.)

A mikroszámítógépek elterjedésével alapvetően megváltozott az átlagember addig kicsit idegenkedő, némi csodálattal fűszerezett viszonya a számítógéphez. Ennek a forradalmi változásnak egyik legszembetűnőbb jele az ember és a számítógép közötti kommunikáció átalakulása. A mikroszámítógép nem alfanumerikus kódokban, hanem közérthető képekben fejezi ki magát.

A számítógépes grafika magában foglalja a grafikai célú eszközök egyre bővülő körét, a készítésükhöz és működtetésükhöz szükséges ismereteket éppúgy, mint a korszerű grafikai programcsomagok és grafikára orientált programnyelvek ismeretét. A számítógépes grafika számtalan területen alkalmazható. Felhasználják a műszaki tervezésben, a tudományos kutatásban

1988–1989-től
LEKÉRDEZHETŐ LESZ
AZ
MTA KÖNYVTÁRA
CD-ROM
(Compact Disc Read-Only Memory)
SZÁMÍTÓGÉPES SZAKIRODALMI
ADATBÁZIS GYŰJTEMÉNYE
lézeroptikai lemezen

kb. 5–10 Giga Byte ⇒ INFORMÁCIÓ ⇐ Többmillió bibliográfiai tétel

ONLINE
AZ INFORMATIKAI SZÁMÍTÓGÉPES HÁLÓZAT ÚTJÁN

OFFLINE
ELŐFIZETÉS ALAPJÁN

KÉZKÖZELBEN
A VILÁG LEGJELENTŐSEBB SZAKIRODALMI
ADATBÁZISAI

ADATBÁZISOK — SZAKIRODALOM (5 sor)

CD-ROM

COMPACT DISC — COMPACT DISC

ONLINE
OFFLINE

SZOLGÁLTATÁS

KÖZVETLENÜL
HÁLÓZATON

Természettudományok
Társadalomtudományok
Orvosbiológia
Online lekérdezés
Offline szolgáltatás
Közvetlenül
a Roosevelt térről
Retrospektív
szakirodalmi keresések

és az oktatás legkülönbözőbb szintjein, térképek készítésében, nem is szólva a számítógépes játékok sokaságáról.

A könyv nem vállalkozik arra, hogy bevezesse az olvasót a számítógépes grafika szerteágazó problémakörébe. Grafikai algoritmusok egy gyűjteményét tartalmazza; olyan matematikai nyelven megfogalmazott, de számítógéppel kivitelezett eljárásokét, amelyeknek célja, hogy valamilyen tárgyat többé-kevésbé hiven ábrázoljanak, vagy hogy érdekes, szép képeket rajzoljanak. Az algoritmusok matematikai apparátust használnak, ami azonban nem lépi túl a középiskolai matematika szintjét.

A könyvben szereplő képek és programok ZX—Spectrumon készültek.

A mikro-számítástechnika 1987. I. félévi piaci helyzete
Összeáll.: Broczkó Péter
(Budapest, 1987.
Központi Statisztikai Hivatal,
60 oldal. Ára: 39,— Ft.)

A kiadvány a magyar mikro-számítástechnikai piac 1987. I. félévi jellemzőit foglalja össze. A hardvertáblázatok a mikroszámítógépek és perifériák árainak alakulását, a szoftvertáblázatok pedig a hazánkban az IBM PC-vel kompatibilis gépekre forgalmazott szoftverárukat tartalmazzák, a forgalmazó cég és az ár feltüntetésével. Mindez elsősorban a mikroszámítógép-beszerzéshez nyújt gyakorlati segítséget.

Intermos

Intermos Mikroelektronikai Kft. néven nemrég magyar—szovjet közös vállalatot hozott létre Budapesten a Mikroelektronikai Vállalat, a Híradástechnika Szövetkezet és két moszkvai intézmény: a Tudományos Központ Termelő-fejlesztő Egyesülés és a Zagranposztavka. Kezdetben a közös vállalat 6-8 fős termeltetőirodát tart fenn, amely a magyar alapító vállalatok kapacitásán számítástechnikai eszközökhöz, mérőműszerekhez gyárt berendezésorientált áramköröket. A gyártáshoz a félkész szeletek a Szovjetunióból érkeznek. A tervek szerint 1989-ben az Intermos már a saját üzemében folytatja a korábbi munkát. A végső cél, hogy a kilencvenes években a teljes technológiai vertikumot az Intermos végezze, azaz a félkész szeleteket is saját maga állítsa elő.

A végterméket eleinte elsősorban a Szovjetunióban értékesítik, de máris megkezdődött mind a hazai, mind a külföldi piacok tájékoztatása az új termékekről.

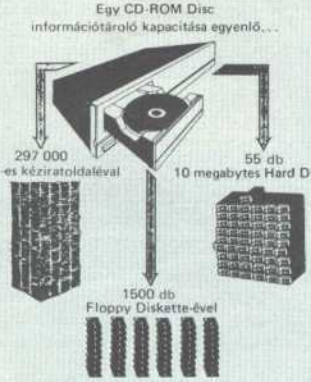
Élő sejtek a képernyőn

Az élettelen anyag belső szerkezete és az élő sejtcellák közvetlenül, minden külső beavatkozás nélkül megfigyelhetők egy új, a számítógép és az elektronmikroszkóp tulajdonságait egyesítő készülékkel. A francia Numelec társaság által kifejlesztett gyors képfeldolgozású analizátorral a korábbi 15 nap helyett néhány óra alatt elvégezhető a szövettani vizsgálatok alapján a rákos daganat elemzése. Hasonlóan rövid idő alatt megfigyelhető a különböző biotechnológiai eljárások során létrehozott baktériumok fejlődése.

A számítógép a mikroszkóp által készített felvételt 256 ezer képpontra bontja fel, úgy tárolja, elemzi, s az eredményt színes képernyőn megjeleníti. A kutató kinagyíthatja a kép egy részét, módosíthatja a színeket a szeme számára megkülönböztethetetlen részek megfigyeléséhez. A gép 256-féle színárnyalatot tud előállítani.

A Magyar Tudományos Akadémia Könyvtára bevezeti az online és offline szakirodalmi információs szolgáltatást a könyvtárban kiépítésre kerülő CD-ROM adatbázis-gyűjteményre alapozva.

A CD-ROM, 12 cm átmérőjű műanyag korong, amely kb. 300–650 MByte információ tárolására alkalmas.



Az MTA Könyvtára CD-ROM adatbázis-gyűjteményét lépcsőzetes kiépítésben számos adatbázis képezi.

Az adatbázisok lekérdezésére a következő lehetőségek állnak majd rendelkezésre:

- Online keresés a MTA Könyvtár Roosevelt téri épületében elhelyezett terminálokon
- Online keresés hálózati kapcsolat alapján saját munkahelyi terminálon
- Offline előfizetés

DATABASES ON CD-ROM HAVE "ARRIVED"

Operational uses of CD-ROM (Compact Disc Read-Only-Memory) technology have skyrocketed, beyond the expectations of most even two or three years ago. CD-ROM has proved practical and economical and is already a significant factor in database availability to Research and Development.

[R & D Management Digest, vol. 15, No. 10, 1986]

Kérem, hogy az MTAK CD-ROM szakirodalmi adatbázis-gyűjtemény fejlesztésével kapcsolatos híreket folyamatosan címemre megküldeni szíveskedjenek:

Név: .....

Munkahely: .....

Postai cím: .....

Kelt: ..... 198 .....

.....
aláírás

Forma—1

Egyre fontosabb szerepet kap a számítógép a Forma—1 világában is. Napjainkban már nem csupán időmérésre, eredmények összehasonlítására, a motorok beállítására használják, hanem hamarosan az egész technika ellenőrzése és irányítása a számítógép dolga lesz. Bombaként robbant a hír a versenytársak között, hogy a Lotus–csapat tavaly a Monacói Nagydíjon elektronikus kerékfelfüggesztést alkalmazott a brazil Senna versenyautóján. Ez az elektronikus érzékelők révén kapott információk alapján úgy irányítja a kocsi futóművét, hogy a 900 lóerős autócsoda útfekvése, úton tartása minden pillanatban optimális.

Nagy Fal Prágában

Kínai számítógép kezdte meg működését a prágai Károly Egyetem matematika—fizika tanszékén. A Nagy Fal elnevezésű, az IBM PC-vel kompatibilis mikrogépet 512 kbájt operatív tárral, két hajlékonylemezes tárolóval, színes képernyővel és mátrixnyomtatóval látták el.

Kicsi a bors...

A világ legkisebb 1 Gbájtos tárolóját fejlesztette ki a japán Hitachi cég. A DK 815-10 típusú tár külső méretei 259 × 216 × 550 mm; a hozzáférést 2,5 Mbájt/s sebességgel biztosítja. A tárolóban 9 db merevlemez van, mellyel 15 db író-olvasó fej dolgozik. A lemezcsomagban lévő cilinderek száma 1537.

Adatvédelem

A számítóközpontok feldolgozás közben kisugárzott információinak védelmét eddig vagy különleges „adatbiztos" számítógépek vásárlásával oldották meg, vagy a védeni kívánt adatokat tartalmazó épületrészek köré különleges házat építettek. Ez leggyakrabban fából készült, belső fémlemezborítással.

Az USA-beli TAFA cég most új, olcsó eljárást kínál: egy szórással felvihető, cinktartalmú festékbevonatot. Ezzel a megfelelő védelmet adó falazatbevonat a lemezekből készült „fémdoboz" költségének egyötödéből előállítható. Az új védőbevonatot már jó néhány vállalati, költségvetési és egyéb számítóközpontban kipróbálták, többek között az FBI számítógép-helyiségeiben is.

Érdiagnózis

A szív- és érrendszeri betegségek megbízhatóbb felismerését segítő orvosi műszerrel bővítik Pécsett a komplex lakossági szűrővizsgálatok eszköztárát. A pécsi és fővárosi szakemberek által létrehozott Hevimet—40 típusú mikrogépes rendszer a betegtől vett vérminta viszkozitásának, vagyis belső súrlódásának mérésére szolgál.

A néhány perces méréssorozat a vér laboratóriumi vizsgálatának eredményeivel együtt minden korábbinál részletesebb tájékoztatást nyújt az érrendszer állapotáról, segítve a halálokok között vezető helyen álló érrendszeri betegségek korai felismerését, illetve megelőzését. Még a dohányzás következtében fellépett egészségkárosodás mértékének pontosabb megállapítását is lehetővé teszi. A műszerrel nyert adatok birtokában az orvos az eddigi általános intelmek helyett konkrét tanácsokkal láthatja el páciensét.

Gabonasiló

A Bólyi Mezőgazdasági Kombinát mikroszámítógéppel vezérelt fémsilótelepet létesített Törökdombon. A közel 10 ezer tonna termény tárolására alkalmas telep tíz fémsilóból és az azokat kiszolgáló technológiai toronyból áll. A beszállított termény attól kezdve, hogy lekerül a teherautóról, teljesen automatikus úton jut a silóba. A vezérlést egy svájci gyártmányú, Saia PCA—2 típusú folyamatirányító mikrogép végzi. A 8 bites, Intel 8085 mikroprocesszor köré épült célgép vezérlőprogramját egy 4 kbájtos EPROM tartalmazza.

A mikrogép alkalmazásának előnye, hogy megnöveli a berendezések élettartamát, kizárja a téves technológiai folyamatokat, villamosenergia-megtakarítást eredményez és megkönnyíti a telep üzemeltetését. A távvezérlés, a távjelzés, az automatikus működés következtében figyelemre méltó az élőmunka-megtakarítás is.

Pontvadászat

Új rejtvényt indítunk útjára lapunkban, remélve, hogy elnyeri olvasóink tetszését. Minden számunkban két feladatot közlünk: az első logikai, matematikai tudást, a második számítástechnikai alapismereteket is igényel.

A feladatok után értékeljük a részmegoldásokat és közöljük az elérhető maximális pontszámot.

A pontgyűjtést, vagyis a pontvadászatot az esztendő végén zárjuk. A legjobb tíz versenyző nevét magazinunkban közzé is tesszük, ők lesznek azok, akik könyvutalványt kapnak.

A helyes megoldások a feladatok közreadása után két lapszámmal később jelennek meg, így a pontvadászoknak jut idejük a gondolkodásra.

Beküldési határidő: 1988. március 14.
Címünk: 1371 Budapest, Pf. 433.

Jó vadászatot kíván a feladatok összeállítója:

dr. Hoffmann Tibor

1. feladat

Tekintsük a 3-mal nem osztható számok sorozatát. Ebből a sorozatból két egymás utáni elem négyzetének a különbsége **568 752.** Melyik ez a két elem? (2 pont) És melyik lenne ez a két elem, ha a különbség **568 755** lenne? (2 pont)

2. feladat

Egy mátrixnyomtató a betűalakokat szélességben **n** pontból és magasságban **m** pontból állítja elő. Melyik az a legkisebb **n** és **m**, mellyel az arab számjegyeket egymástól jól megkülönböztethetően elő tudjuk állítani? (3 pont)

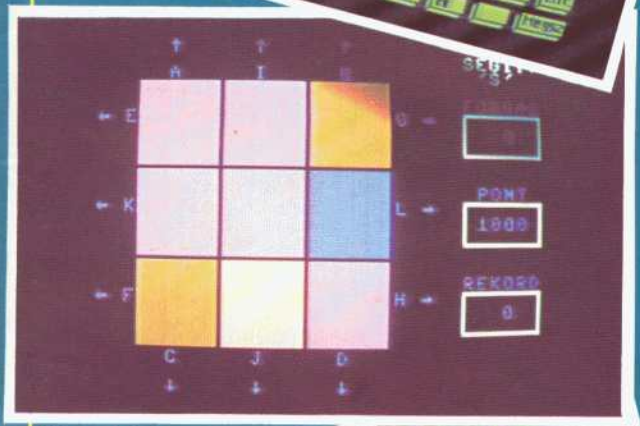
Hány pontból állíthatók így elő az egyes **0** és **9** közötti számok? (1 pont)

Ennek következtében melyik számmal célszerű nyomtatótesztet végezni? (1 pont)

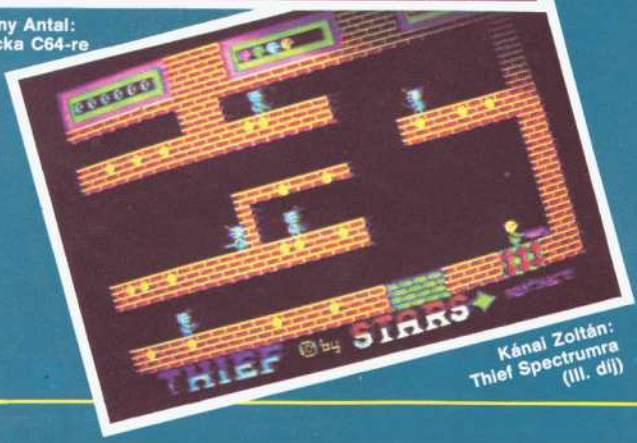
Software '88
JÁTÉKPÁLYÁZAT

Ők nyertek!

Rozsnyai György:
Spectmusic Spectrumra
(I. díj, zenei különdíj)



Dékány Antal:
A kocka C64-re



Kánai Zoltán:
Thief Spectrumra
(III. díj)

A Hotel Duna Intercontinentalban megrendezett Software '88 reprezentatív kiállításon ünnepélyes keretek között adták közre a játékpályázat nyerteseinek névsorát. A KISZ KB, az ÁISH által működtetett Ifjúsági Műszaki Központ és a SZÁMALK közös felhívására 59 pályamunka érkezett, s ezek közül választotta ki az avatott zsűri a legjobbakat.

Első lett Rozsnyai György (az ELTE TTK programozómatematikus szak előfelvételise, jelenleg sorkatona), aki ZX Spectrumra írta győztes Spectmusic programját. Zenekomponálásra, tárolásra és tökéletesítésre alkalmas. A pályamunka érdekes, ötletes, s többhónapos programozás eredménye. Egyébként Rozsnyai György ezzel a programmal elnyerte a zsűri zenei különdíját is.

Második helyezett lett Dékány Antal (pécsi mérnök), a Rubik-kocka forgatását vitte számítógépre. Véletlenszerűen állítja elő a program a kocka 6 lapjának színkeverését a C64-es gépen.

A harmadik díjat megosztva adta ki a zsűri, Nagy László (budapesti programozó) C64-re íródott Puzzle képösszerakó játékával nyert, Kánai Zoltán (gyáli középiskolás) ZX Spectrumra készített Talicskás nevű játékot. Ennek technikai megoldása egyedi és ötletes.

A zsűri másik különdíját a grafikai munkáért Kosir Attila (budapesti tanuló) nyerte. A közönségdíjat az a program kapja, amelyet 1988. szeptember 30-ig hazánkban a legnagyobb példányszámban értékesítenek.

Négy játékötletet díjaztak. Kovács Levente, Mikó András és a Szent Jupát, illetve az Apacs jeligével pályázó ismeretlenek nyertek.

A beérkezett pályamunkákat összegezve elmondható, hogy kiugró teljesítményű és újdonságnak számító program csak az első helyezettől érkezett. Ugyanakkor mintegy 30 program forgalmazását tervezi a SZÁMALK és viszonteladói. Többek között a Művelt Nép Könyvterjesztő Vállalat számítástechnikai szakboltjaiban, valamint az ÁPISZ—SZÁMALK közös üzemeltetésű boltjaiban árusítják majd. A tervek szerint 249—299 forintért forgalmazzák a programokat. Egy kazettán, illetve floppyn több játék is lesz. A SZÁMALK a szerzők közreműködésével tovább kívánja fejleszteni a programokat, hogy azok alkalmasak legyenek nyugat-európai értékesítésre is.

Software '88
JÁTÉKPÁLYÁZAT

Kosir Attila:
Godzilla Plus 4-re
(grafikai díj)

Fenyvesi Béla:
Lufik az űrben C64-re

Király Péterné:
Összerakó játék és Memória
Spectrumra